# Gazebo

2.2.1

Generated by Doxygen 1.8.3.1

Thu Jan 23 2014 11:18:45

# Contents


**1 Gazebo API Reference** **1**

**2 Todo List** **3**

**3 Module Index** **5**

3.1 Modules . . . . . 5

**4 Namespace Index** **7**

4.1 Namespace List . . . . . 7

**5 Hierarchical Index** **9**

5.1 Class Hierarchy . . . . . 9

**6 Class Index** **17**

6.1 Class List . . . . . 17

**7 File Index** **27**

7.1 File List . . . . . 27

**8 Module Documentation** **33**

8.1 Common . . . . . 33

8.1.1 Detailed Description . . . . . 37

8.1.2 Macro Definition Documentation . . . . . 37

8.1.2.1 gzclr_end . . . . . 37

8.1.2.2 gzclr_start . . . . . 37

8.1.2.3 gzdbg . . . . . 37

8.1.2.4 gzerr . . . . . 37

8.1.2.5 gzlog . . . . . 38

8.1.2.6 gzmsg . . . . . 38

8.1.2.7 gzthrow . . . . . 38

8.1.2.8 gzwarn . . . . . 38

8.1.3 Enumeration Type Documentation . . . 38
8.1.3.1 PluginType . . . 38
8.1.4 Function Documentation . . . 39
8.1.4.1 NullStream . . . 39
8.1.4.2 add_search_path_suffix . . . 39
8.1.4.3 ColorErr . . . 39
8.1.4.4 ColorMsg . . . 39
8.1.4.5 DownloadDependencies . . . 39
8.1.4.6 find_file . . . 40
8.1.4.7 find_file . . . 40
8.1.4.8 find_file_path . . . 40
8.1.4.9 Fini . . . 40
8.1.4.10 get_sha1 . . . 40
8.1.4.11 GetDBConfig . . . 41
8.1.4.12 GetModelConfig . . . 41
8.1.4.13 GetModelFile . . . 41
8.1.4.14 GetModelName . . . 41
8.1.4.15 GetModelPath . . . 42
8.1.4.16 GetModels . . . 42
8.1.4.17 GetModels . . . 42
8.1.4.18 GetQuiet . . . 42
8.1.4.19 GetURI . . . 42
8.1.4.20 HasModel . . . 43
8.1.4.21 Init . . . 43
8.1.4.22 IsInitialized . . . 43
8.1.4.23 load . . . 43
8.1.4.24 Log . . . 43
8.1.4.25 SetQuiet . . . 43
8.1.4.26 Start . . . 44
8.1.5 Variable Documentation . . . 44
8.1.5.1 PixelFormatNames . . . 44
8.2 Events . . . 45
8.2.1 Detailed Description . . . 45
8.2.2 Function Documentation . . . 45
8.2.2.1 ∼EventT . . . 45
8.2.2.2 Connect . . . 45
8.2.2.3 ConnectionCount . . . 46

8.2.2.4 Disconnect . . . 46
8.2.2.5 Disconnect . . . 47
8.3 Math . . . 48
8.3.1 Detailed Description . . . 49
8.3.2 Function Documentation . . . 50
8.3.2.1 clamp . . . 50
8.3.2.2 equal . . . 50
8.3.2.3 fixnan . . . 50
8.3.2.4 fixnan . . . 50
8.3.2.5 isnan . . . 51
8.3.2.6 isnan . . . 51
8.3.2.7 isPowerOfTwo . . . 51
8.3.2.8 max . . . 51
8.3.2.9 mean . . . 52
8.3.2.10 min . . . 52
8.3.2.11 parseFloat . . . 52
8.3.2.12 parseInt . . . 53
8.3.2.13 precision . . . 53
8.3.2.14 variance . . . 53
8.3.3 Variable Documentation . . . 53
8.3.3.1 NAN_D . . . 53
8.3.3.2 NAN_I . . . 53
8.4 Messages . . . 54
8.4.1 Detailed Description . . . 56
8.4.2 Macro Definition Documentation . . . 56
8.4.2.1 GZ_REGISTER_STATIC_MSG . . . 56
8.4.3 Function Documentation . . . 56
8.4.3.1 Convert . . . 56
8.4.3.2 Convert . . . 56
8.4.3.3 Convert . . . 57
8.4.3.4 Convert . . . 57
8.4.3.5 Convert . . . 57
8.4.3.6 Convert . . . 57
8.4.3.7 Convert . . . 58
8.4.3.8 Convert . . . 58
8.4.3.9 Convert . . . 58
8.4.3.10 Convert . . . 58

8.4.3.11 Convert . . . 59
8.4.3.12 Convert . . . 59
8.4.3.13 CreateRequest . . . 59
8.4.3.14 FogFromSDF . . . 59
8.4.3.15 GeometryFromSDF . . . 60
8.4.3.16 GetHeader . . . 60
8.4.3.17 GUIFromSDF . . . 60
8.4.3.18 Init . . . 60
8.4.3.19 LightFromSDF . . . 61
8.4.3.20 MeshFromSDF . . . 61
8.4.3.21 SceneFromSDF . . . 61
8.4.3.22 Set . . . 61
8.4.3.23 Set . . . 62
8.4.3.24 Set . . . 62
8.4.3.25 Set . . . 62
8.4.3.26 Set . . . 62
8.4.3.27 Set . . . 62
8.4.3.28 Set . . . 62
8.4.3.29 Set . . . 63
8.4.3.30 Set . . . 63
8.4.3.31 Stamp . . . 63
8.4.3.32 Stamp . . . 63
8.4.3.33 TrackVisualFromSDF . . . 63
8.4.3.34 VisualFromSDF . . . 64
8.5 Classes for physics and dynamics . . . 65
8.5.1 Detailed Description . . . 69
8.5.2 Macro Definition Documentation . . . 69
8.5.2.1 GZ_REGISTER_PHYSICS_ENGINE . . . 69
8.5.3 Typedef Documentation . . . 69
8.5.3.1 PhysicsFactoryFn . . . 69
8.5.4 Function Documentation . . . 69
8.5.4.1 create_world . . . 69
8.5.4.2 fini . . . 69
8.5.4.3 get_world . . . 69
8.5.4.4 getUniqueId . . . 70
8.5.4.5 init_world . . . 70
8.5.4.6 init_worlds . . . 70

8.5.4.7 load . . . 70
8.5.4.8 load_world . . . 70
8.5.4.9 load_worlds . . . 70
8.5.4.10 pause_world . . . 71
8.5.4.11 pause_worlds . . . 71
8.5.4.12 remove_worlds . . . 71
8.5.4.13 run_world . . . 71
8.5.4.14 run_worlds . . . 71
8.5.4.15 stop_world . . . 71
8.5.4.16 stop_worlds . . . 72
8.5.4.17 worlds_running . . . 72
8.5.5 Variable Documentation . . . 72
8.5.5.1 EntityTypename . . . 72
8.6 DART Physics . . . 73
8.6.1 Detailed Description . . . 74
8.6.2 Function Documentation . . . 74
8.6.2.1 DARTRayShape . . . 74
8.6.2.2 DARTRayShape . . . 74
8.6.2.3 ∼DARTRayShape . . . 74
8.6.2.4 ConvPose . . . 74
8.6.2.5 ConvPose . . . 74
8.6.2.6 ConvQuat . . . 75
8.6.2.7 ConvQuat . . . 75
8.6.2.8 ConvVec3 . . . 75
8.6.2.9 ConvVec3 . . . 75
8.6.2.10 GetIntersection . . . 75
8.6.2.11 SetPoints . . . 75
8.6.2.12 Update . . . 75
8.7 Bullet Physics . . . 76
8.7.1 Detailed Description . . . 76
8.7.2 Function Documentation . . . 76
8.7.2.1 DARTMultiRayShape . . . 76
8.7.2.2 ∼DARTMultiRayShape . . . 76
8.7.2.3 AddRay . . . 77
8.7.2.4 UpdateRays . . . 77
8.8 Simbody Physics . . . 78
8.8.1 Detailed Description . . . 79

8.9 Rendering . . . 80
8.9.1 Detailed Description . . . 83
8.9.2 Enumeration Type Documentation . . . 83
8.9.2.1 SelectionMode . . . 83
8.9.3 Function Documentation . . . 83
8.9.3.1 SelectionObj . . . 83
8.9.3.2 ∼SelectionObj . . . 83
8.9.3.3 Attach . . . 84
8.9.3.4 create_scene . . . 84
8.9.3.5 Detach . . . 84
8.9.3.6 fini . . . 84
8.9.3.7 get_scene . . . 84
8.9.3.8 GetMode . . . 84
8.9.3.9 GetState . . . 84
8.9.3.10 init . . . 84
8.9.3.11 load . . . 85
8.9.3.12 Load . . . 85
8.9.3.13 remove_scene . . . 85
8.9.3.14 SetGlobal . . . 85
8.9.3.15 SetMode . . . 85
8.9.3.16 SetMode . . . 85
8.9.3.17 SetState . . . 85
8.9.3.18 SetState . . . 86
8.9.3.19 UpdateSize . . . 86
8.10 Sensors . . . 87
8.10.1 Detailed Description . . . 89
8.10.2 Macro Definition Documentation . . . 89
8.10.2.1 GZ_REGISTER_STATIC_SENSOR . . . 89
8.10.3 Function Documentation . . . 89
8.10.3.1 create_sensor . . . 89
8.10.3.2 create_sensor . . . 89
8.10.3.3 disable . . . 90
8.10.3.4 enable . . . 90
8.10.3.5 fini . . . 90
8.10.3.6 get_sensor . . . 90
8.10.3.7 init . . . 90
8.10.3.8 load . . . 90

8.10.3.9 remove_sensor . . . 91
8.10.3.10 remove_sensors . . . 91
8.10.3.11 run_once . . . 91
8.10.3.12 run_threads . . . 91
8.10.3.13 stop . . . 91
8.11 Transport . . . 92
8.11.1 Detailed Description . . . 94
8.11.2 Typedef Documentation . . . 94
8.11.2.1 CallbackHelperPtr . . . 94
8.11.3 Function Documentation . . . 94
8.11.3.1 clear_buffers . . . 94
8.11.3.2 fini . . . 94
8.11.3.3 get_master_uri . . . 94
8.11.3.4 get_topic_namespaces . . . 94
8.11.3.5 getAdvertisedTopics . . . 95
8.11.3.6 getAdvertisedTopics . . . 95
8.11.3.7 getMinimalComms . . . 95
8.11.3.8 getTopicMsgType . . . 95
8.11.3.9 init . . . 95
8.11.3.10 is_stopped . . . 96
8.11.3.11 pause_incoming . . . 96
8.11.3.12 publish . . . 96
8.11.3.13 request . . . 96
8.11.3.14 requestNoReply . . . 97
8.11.3.15 requestNoReply . . . 97
8.11.3.16 run . . . 97
8.11.3.17 setMinimalComms . . . 97
8.11.3.18 stop . . . 97
8.12 Utility . . . 98
8.12.1 Detailed Description . . . 98
8.12.2 Macro Definition Documentation . . . 98
8.12.2.1 DIAG_TIMER_LAP . . . 98
8.12.2.2 DIAG_TIMER_START . . . 98
8.12.2.3 DIAG_TIMER_STOP . . . 98

**9 Namespace Documentation** **99**

9.1 boost Namespace Reference . . . 99

9.2 gazebo Namespace Reference . . . 99
9.2.1 Detailed Description . . . 100
9.2.2 Typedef Documentation . . . 101
9.2.2.1 GUIPluginPtr . . . 101
9.2.2.2 ModelPluginPtr . . . 101
9.2.2.3 SensorPluginPtr . . . 101
9.2.2.4 SystemPluginPtr . . . 101
9.2.2.5 VisualPluginPtr . . . 101
9.2.2.6 WorldPluginPtr . . . 101
9.2.3 Function Documentation . . . 101
9.2.3.1 add_plugin . . . 101
9.2.3.2 find_file . . . 101
9.2.3.3 fini . . . 101
9.2.3.4 init . . . 101
9.2.3.5 load . . . 101
9.2.3.6 print_version . . . 101
9.2.3.7 run . . . 101
9.2.3.8 stop . . . 101
9.3 gazebo::common Namespace Reference . . . 101
9.3.1 Detailed Description . . . 104
9.3.2 Typedef Documentation . . . 104
9.3.2.1 AnimationPtr . . . 104
9.3.2.2 DiagnosticTimerPtr . . . 104
9.3.2.3 NodeMap . . . 104
9.3.2.4 NodeMapIter . . . 104
9.3.2.5 NumericAnimationPtr . . . 104
9.3.2.6 Param_V . . . 104
9.3.2.7 PoseAnimationPtr . . . 104
9.3.2.8 RawNodeAnim . . . 104
9.3.2.9 RawNodeWeights . . . 104
9.3.2.10 RawSkeletonAnim . . . 104
9.3.2.11 SphericalCoordinatesPtr . . . 105
9.3.2.12 StrStr_M . . . 105
9.3.3 Variable Documentation . . . 105
9.3.3.1 SpeedOfLight . . . 105
9.4 gazebo::event Namespace Reference . . . 105
9.4.1 Detailed Description . . . 105

9.4.2 Typedef Documentation . . . 105
9.4.2.1 Connection_V . . . 105
9.4.2.2 ConnectionPtr . . . 105
9.5 gazebo::math Namespace Reference . . . 105
9.5.1 Detailed Description . . . 108
9.5.2 Typedef Documentation . . . 108
9.5.2.1 GeneratorType . . . 108
9.5.2.2 NormalRealDist . . . 108
9.5.2.3 NRealGen . . . 108
9.5.2.4 UIntGen . . . 108
9.5.2.5 UniformIntDist . . . 108
9.5.2.6 UniformRealDist . . . 108
9.5.2.7 URealGen . . . 108
9.6 gazebo::msgs Namespace Reference . . . 108
9.6.1 Detailed Description . . . 110
9.6.2 Typedef Documentation . . . 110
9.6.2.1 MsgFactoryFn . . . 110
9.7 gazebo::physics Namespace Reference . . . 110
9.7.1 Detailed Description . . . 116
9.7.2 Typedef Documentation . . . 116
9.7.2.1 Actor_V . . . 116
9.7.2.2 ActorPtr . . . 116
9.7.2.3 Base_V . . . 116
9.7.2.4 BasePtr . . . 116
9.7.2.5 BoxShapePtr . . . 116
9.7.2.6 Collision_V . . . 116
9.7.2.7 CollisionPtr . . . 116
9.7.2.8 ContactPtr . . . 116
9.7.2.9 CylinderShapePtr . . . 116
9.7.2.10 DARTCollisionPtr . . . 116
9.7.2.11 DARTJointPtr . . . 116
9.7.2.12 DARTLinkPtr . . . 116
9.7.2.13 DARTModelPtr . . . 117
9.7.2.14 DARTPhysicsPtr . . . 117
9.7.2.15 DARTRayShapePtr . . . 117
9.7.2.16 EntityPtr . . . 117
9.7.2.17 GripperPtr . . . 117

9.7.2.18 HeightmapShapePtr . . . 117
9.7.2.19 InertialPtr . . . 117
9.7.2.20 Joint_V . . . 117
9.7.2.21 JointController_V . . . 117
9.7.2.22 JointControllerPtr . . . 117
9.7.2.23 JointPtr . . . 117
9.7.2.24 JointState_M . . . 117
9.7.2.25 Link_V . . . 117
9.7.2.26 LinkPtr . . . 117
9.7.2.27 LinkState_M . . . 117
9.7.2.28 MeshShapePtr . . . 117
9.7.2.29 Model_V . . . 117
9.7.2.30 ModelPtr . . . 117
9.7.2.31 ModelState_M . . . 117
9.7.2.32 MultiRayShapePtr . . . 117
9.7.2.33 PhysicsEnginePtr . . . 117
9.7.2.34 RayShapePtr . . . 117
9.7.2.35 RoadPtr . . . 117
9.7.2.36 ShapePtr . . . 117
9.7.2.37 SimbodyCollisionPtr . . . 117
9.7.2.38 SimbodyLinkPtr . . . 117
9.7.2.39 SimbodyModelPtr . . . 118
9.7.2.40 SimbodyPhysicsPtr . . . 118
9.7.2.41 SimbodyRayShapePtr . . . 118
9.7.2.42 SphereShapePtr . . . 118
9.7.2.43 SurfaceParamsPtr . . . 118
9.7.2.44 WorldPtr . . . 118
9.8 gazebo::rendering Namespace Reference . . . 118
9.8.1 Detailed Description . . . 121
9.8.2 Typedef Documentation . . . 121
9.8.2.1 ArrowVisualPtr . . . 121
9.8.2.2 AxisVisualPtr . . . 121
9.8.2.3 CameraPtr . . . 121
9.8.2.4 CameraVisualPtr . . . 121
9.8.2.5 COMVisualPtr . . . 121
9.8.2.6 ContactVisualPtr . . . 121
9.8.2.7 DepthCameraPtr . . . 121

9.8.2.8 DynamicLinesPtr . . . 121
9.8.2.9 GpuLaserPtr . . . 121
9.8.2.10 JointVisualPtr . . . 121
9.8.2.11 LaserVisualPtr . . . 121
9.8.2.12 LightPtr . . . 121
9.8.2.13 RFIDTagVisualPtr . . . 121
9.8.2.14 RFIDVisualPtr . . . 121
9.8.2.15 ScenePtr . . . 122
9.8.2.16 SelectionObjPtr . . . 122
9.8.2.17 SonarVisualPtr . . . 122
9.8.2.18 UserCameraPtr . . . 122
9.8.2.19 VisualPtr . . . 122
9.8.2.20 WindowManagerPtr . . . 122
9.8.2.21 WrenchVisualPtr . . . 122
9.8.3 Enumeration Type Documentation . . . 122
9.8.3.1 RenderOpType . . . 122
9.9 gazebo::sensors Namespace Reference . . . 122
9.9.1 Detailed Description . . . 125
9.9.2 Typedef Documentation . . . 125
9.9.2.1 CameraSensor_V . . . 125
9.9.2.2 CameraSensorPtr . . . 125
9.9.2.3 ContactSensor_V . . . 125
9.9.2.4 ContactSensorPtr . . . 125
9.9.2.5 DepthCameraSensor_V . . . 125
9.9.2.6 DepthCameraSensorPtr . . . 125
9.9.2.7 ForceTorqueSensorPtr . . . 125
9.9.2.8 GpsSensorPtr . . . 125
9.9.2.9 GpuRaySensor_V . . . 125
9.9.2.10 GpuRaySensorPtr . . . 125
9.9.2.11 ImuSensor_V . . . 125
9.9.2.12 ImuSensorPtr . . . 125
9.9.2.13 MultiCameraSensor_V . . . 126
9.9.2.14 MultiCameraSensorPtr . . . 126
9.9.2.15 NoisePtr . . . 126
9.9.2.16 RaySensor_V . . . 126
9.9.2.17 RaySensorPtr . . . 126
9.9.2.18 RFIDSensor_V . . . 126

9.9.2.19 RFIDSensorPtr . . . 126
9.9.2.20 RFIDTag_V . . . 126
9.9.2.21 RFIDTagPtr . . . 126
9.9.2.22 Sensor_V . . . 126
9.9.2.23 SensorFactoryFn . . . 126
9.9.2.24 SensorPtr . . . 126
9.9.2.25 SonarSensorPtr . . . 126
9.9.2.26 WirelessReceiver_V . . . 126
9.9.2.27 WirelessReceiverPtr . . . 126
9.9.2.28 WirelessTransceiver_V . . . 126
9.9.2.29 WirelessTransceiverPtr . . . 126
9.9.2.30 WirelessTransmitter_V . . . 126
9.9.2.31 WirelessTransmitterPtr . . . 126
9.9.3 Enumeration Type Documentation . . . 126
9.9.3.1 SensorCategory . . . 126
9.10 gazebo::transport Namespace Reference . . . 127
9.10.1 Typedef Documentation . . . 129
9.10.1.1 ConnectionPtr . . . 129
9.10.1.2 MessagePtr . . . 129
9.10.1.3 NodePtr . . . 129
9.10.1.4 PublicationPtr . . . 129
9.10.1.5 PublicationTransportPtr . . . 129
9.10.1.6 PublisherPtr . . . 129
9.10.1.7 SubscriberPtr . . . 129
9.10.1.8 SubscriptionTransportPtr . . . 129
9.11 gazebo::util Namespace Reference . . . 129
9.11.1 Typedef Documentation . . . 129
9.11.1.1 DiagnosticTimerPtr . . . 129
9.11.1.2 OpenALSinkPtr . . . 130
9.11.1.3 OpenALSourcePtr . . . 130
9.12 google Namespace Reference . . . 130
9.13 google::protobuf Namespace Reference . . . 130
9.14 google::protobuf::compiler Namespace Reference . . . 130
9.15 google::protobuf::compiler::cpp Namespace Reference . . . 130
9.16 Ogre Namespace Reference . . . 130
9.17 ogre Namespace Reference . . . 130
9.18 SimTK Namespace Reference . . . 130

9.19 SkyX Namespace Reference . . . 130

**10 Class Documentation** **131**

10.1 gazebo::physics::Actor Class Reference . . . 131

10.1.1 Detailed Description . . . 133

10.1.2 Constructor & Destructor Documentation . . . 133

10.1.2.1 Actor . . . 133

10.1.2.2 ∼Actor . . . 134

10.1.3 Member Function Documentation . . . 134

10.1.3.1 Fini . . . 134

10.1.3.2 GetSDF . . . 134

10.1.3.3 Init . . . 134

10.1.3.4 IsActive . . . 134

10.1.3.5 Load . . . 134

10.1.3.6 Play . . . 134

10.1.3.7 Stop . . . 135

10.1.3.8 Update . . . 135

10.1.3.9 UpdateParameters . . . 135

10.1.4 Member Data Documentation . . . 135

10.1.4.1 active . . . 135

10.1.4.2 autoStart . . . 135

10.1.4.3 bonePosePub . . . 135

10.1.4.4 interpolateX . . . 135

10.1.4.5 lastPos . . . 135

10.1.4.6 lastScriptTime . . . 135

10.1.4.7 lastTraj . . . 136

10.1.4.8 loop . . . 136

10.1.4.9 mainLink . . . 136

10.1.4.10 mesh . . . 136

10.1.4.11 oldAction . . . 136

10.1.4.12 pathLength . . . 136

10.1.4.13 playStartTime . . . 136

10.1.4.14 prevFrameTime . . . 136

10.1.4.15 scriptLength . . . 136

10.1.4.16 skelAnimation . . . 136

10.1.4.17 skeleton . . . 136

10.1.4.18 skelNodesMap . . . 137

10.1.4.19 skinFile . . . 137
10.1.4.20 skinScale . . . 137
10.1.4.21 startDelay . . . 137
10.1.4.22 trajectories . . . 137
10.1.4.23 trajInfo . . . 137
10.1.4.24 visualId . . . 137
10.1.4.25 visualName . . . 137
10.2 gazebo::math::Angle Class Reference . . . 137
10.2.1 Detailed Description . . . 139
10.2.2 Constructor & Destructor Documentation . . . 139
10.2.2.1 Angle . . . 139
10.2.2.2 Angle . . . 139
10.2.2.3 Angle . . . 139
10.2.2.4 ∼Angle . . . 139
10.2.3 Member Function Documentation . . . 140
10.2.3.1 Degree . . . 140
10.2.3.2 Normalize . . . 140
10.2.3.3 operator!= . . . 140
10.2.3.4 operator∗ . . . 140
10.2.3.5 operator∗ . . . 140
10.2.3.6 operator∗= . . . 140
10.2.3.7 operator+ . . . 141
10.2.3.8 operator+= . . . 141
10.2.3.9 operator- . . . 141
10.2.3.10 operator-= . . . 141
10.2.3.11 operator/ . . . 142
10.2.3.12 operator/= . . . 142
10.2.3.13 operator< . . . 142
10.2.3.14 operator<= . . . 142
10.2.3.15 operator== . . . 143
10.2.3.16 operator> . . . 143
10.2.3.17 operator>= . . . 143
10.2.3.18 Radian . . . 143
10.2.3.19 SetFromDegree . . . 144
10.2.3.20 SetFromRadian . . . 144
10.2.4 Friends And Related Function Documentation . . . 144
10.2.4.1 operator<< . . . 144

10.2.4.2 operator>> . . . 144
10.2.5 Member Data Documentation . . . 145
10.2.5.1 HalfPi . . . 145
10.2.5.2 Pi . . . 145
10.2.5.3 TwoPi . . . 145
10.2.5.4 Zero . . . 145
10.3 gazebo::common::Animation Class Reference . . . 145
10.3.1 Detailed Description . . . 146
10.3.2 Member Typedef Documentation . . . 147
10.3.2.1 KeyFrame_V . . . 147
10.3.3 Constructor & Destructor Documentation . . . 147
10.3.3.1 Animation . . . 147
10.3.3.2 ~Animation . . . 147
10.3.4 Member Function Documentation . . . 147
10.3.4.1 AddTime . . . 147
10.3.4.2 GetKeyFrame . . . 147
10.3.4.3 GetKeyFrameCount . . . 147
10.3.4.4 GetKeyFramesAtTime . . . 148
10.3.4.5 GetLength . . . 148
10.3.4.6 GetTime . . . 148
10.3.4.7 SetLength . . . 148
10.3.4.8 SetTime . . . 148
10.3.5 Member Data Documentation . . . 149
10.3.5.1 build . . . 149
10.3.5.2 keyFrames . . . 149
10.3.5.3 length . . . 149
10.3.5.4 loop . . . 149
10.3.5.5 name . . . 149
10.3.5.6 timePos . . . 149
10.4 gazebo::rendering::ArrowVisual Class Reference . . . 149
10.4.1 Detailed Description . . . 150
10.4.2 Constructor & Destructor Documentation . . . 150
10.4.2.1 ArrowVisual . . . 150
10.4.2.2 ~ArrowVisual . . . 151
10.4.3 Member Function Documentation . . . 151
10.4.3.1 Load . . . 151
10.4.3.2 ShowRotation . . . 151

10.5 gazebo::common::AssertionInternalError Class Reference . . . 151
10.5.1 Detailed Description . . . 152
10.5.2 Constructor & Destructor Documentation . . . 152
10.5.2.1 AssertionInternalError . . . 152
10.5.2.2 ∼AssertionInternalError . . . 153
10.6 gazebo::common::AudioDecoder Class Reference . . . 153
10.6.1 Detailed Description . . . 153
10.6.2 Constructor & Destructor Documentation . . . 153
10.6.2.1 AudioDecoder . . . 153
10.6.2.2 ∼AudioDecoder . . . 153
10.6.3 Member Function Documentation . . . 154
10.6.3.1 Decode . . . 154
10.6.3.2 GetFile . . . 154
10.6.3.3 GetSampleRate . . . 154
10.6.3.4 SetFile . . . 154
10.7 gazebo::rendering::AxisVisual Class Reference . . . 155
10.7.1 Detailed Description . . . 156
10.7.2 Constructor & Destructor Documentation . . . 156
10.7.2.1 AxisVisual . . . 156
10.7.2.2 ∼AxisVisual . . . 156
10.7.3 Member Function Documentation . . . 156
10.7.3.1 Load . . . 156
10.7.3.2 ScaleXAxis . . . 156
10.7.3.3 ScaleYAxis . . . 156
10.7.3.4 ScaleZAxis . . . 157
10.7.3.5 SetAxisMaterial . . . 157
10.7.3.6 ShowRotation . . . 157
10.8 gazebo::physics::BallJoint< T > Class Template Reference . . . 157
10.8.1 Detailed Description . . . 158
10.8.2 Constructor & Destructor Documentation . . . 158
10.8.2.1 BallJoint . . . 158
10.8.2.2 ∼BallJoint . . . 158
10.8.3 Member Function Documentation . . . 158
10.8.3.1 GetAngleCount . . . 158
10.8.3.2 GetHighStop . . . 158
10.8.3.3 GetLowStop . . . 158
10.8.3.4 Load . . . 158

10.8.3.5 SetAxis . . . 159
10.8.3.6 SetHighStop . . . 159
10.8.3.7 SetLowStop . . . 159
10.9 gazebo::physics::Base Class Reference . . . 159
10.9.1 Detailed Description . . . 163
10.9.2 Member Enumeration Documentation . . . 163
10.9.2.1 EntityType . . . 163
10.9.3 Constructor & Destructor Documentation . . . 164
10.9.3.1 Base . . . 164
10.9.3.2 ∼Base . . . 164
10.9.4 Member Function Documentation . . . 164
10.9.4.1 AddChild . . . 164
10.9.4.2 AddType . . . 164
10.9.4.3 ComputeScopedName . . . 164
10.9.4.4 Fini . . . 165
10.9.4.5 GetByName . . . 165
10.9.4.6 GetChild . . . 165
10.9.4.7 GetChild . . . 165
10.9.4.8 GetChildCount . . . 165
10.9.4.9 GetId . . . 166
10.9.4.10 GetName . . . 166
10.9.4.11 GetParent . . . 166
10.9.4.12 GetParentId . . . 166
10.9.4.13 GetSaveable . . . 166
10.9.4.14 GetScopedName . . . 167
10.9.4.15 GetSDF . . . 167
10.9.4.16 GetType . . . 167
10.9.4.17 GetWorld . . . 167
10.9.4.18 HasType . . . 167
10.9.4.19 Init . . . 167
10.9.4.20 IsSelected . . . 168
10.9.4.21 Load . . . 168
10.9.4.22 operator== . . . 169
10.9.4.23 Print . . . 169
10.9.4.24 RemoveChild . . . 169
10.9.4.25 RemoveChild . . . 169
10.9.4.26 RemoveChildren . . . 169

10.9.4.27 Reset . . . 169
10.9.4.28 Reset . . . 170
10.9.4.29 SetName . . . 170
10.9.4.30 SetParent . . . 170
10.9.4.31 SetSaveable . . . 170
10.9.4.32 SetSelected . . . 170
10.9.4.33 SetWorld . . . 170
10.9.4.34 Update . . . 171
10.9.4.35 UpdateParameters . . . 171
10.9.5 Member Data Documentation . . . 171
10.9.5.1 children . . . 171
10.9.5.2 childrenEnd . . . 171
10.9.5.3 parent . . . 171
10.9.5.4 sdf . . . 171
10.9.5.5 world . . . 171
10.10gazebo::math::Box Class Reference . . . 172
10.10.1 Detailed Description . . . 173
10.10.2 Constructor & Destructor Documentation . . . 173
10.10.2.1 Box . . . 173
10.10.2.2 Box . . . 173
10.10.2.3 Box . . . 173
10.10.2.4 ∼Box . . . 173
10.10.3 Member Function Documentation . . . 173
10.10.3.1 GetCenter . . . 173
10.10.3.2 GetSize . . . 174
10.10.3.3 GetXLength . . . 174
10.10.3.4 GetYLength . . . 174
10.10.3.5 GetZLength . . . 174
10.10.3.6 Merge . . . 174
10.10.3.7 operator+ . . . 174
10.10.3.8 operator+= . . . 175
10.10.3.9 operator- . . . 175
10.10.3.10operator= . . . 175
10.10.3.11operator== . . . 175
10.10.4 Friends And Related Function Documentation . . . 176
10.10.4.1 operator<< . . . 176
10.10.5 Member Data Documentation . . . 176

10.10.5.1 max . . . 176
10.10.5.2 min . . . 176
10.11 gazebo::physics::BoxShape Class Reference . . . 176
10.11.1 Detailed Description . . . 178
10.11.2 Constructor & Destructor Documentation . . . 178
10.11.2.1 BoxShape . . . 178
10.11.2.2 ∼BoxShape . . . 178
10.11.3 Member Function Documentation . . . 178
10.11.3.1 FillMsg . . . 178
10.11.3.2 GetSize . . . 178
10.11.3.3 Init . . . 178
10.11.3.4 ProcessMsg . . . 178
10.11.3.5 SetScale . . . 179
10.11.3.6 SetSize . . . 179
10.12 gazebo::common::BVHLoader Class Reference . . . 179
10.12.1 Detailed Description . . . 179
10.12.2 Constructor & Destructor Documentation . . . 180
10.12.2.1 BVHLoader . . . 180
10.12.2.2 ∼BVHLoader . . . 180
10.12.3 Member Function Documentation . . . 180
10.12.3.1 Load . . . 180
10.13 gazebo::transport::CallbackHelper Class Reference . . . 180
10.13.1 Detailed Description . . . 181
10.13.2 Constructor & Destructor Documentation . . . 182
10.13.2.1 CallbackHelper . . . 182
10.13.2.2 ∼CallbackHelper . . . 182
10.13.3 Member Function Documentation . . . 182
10.13.3.1 GetId . . . 182
10.13.3.2 GetLatching . . . 182
10.13.3.3 GetMsgType . . . 182
10.13.3.4 HandleData . . . 182
10.13.3.5 HandleMessage . . . 183
10.13.3.6 IsLocal . . . 183
10.13.4 Member Data Documentation . . . 183
10.13.4.1 latching . . . 183
10.14 gazebo::transport::CallbackHelperT< M > Class Template Reference . . . 184
10.14.1 Detailed Description . . . 184

10.14.2 Constructor & Destructor Documentation . . . 184
10.14.2.1 CallbackHelperT . . . 185
10.14.3 Member Function Documentation . . . 185
10.14.3.1 GetMsgType . . . 185
10.14.3.2 HandleData . . . 185
10.14.3.3 HandleMessage . . . 185
10.14.3.4 IsLocal . . . 186
10.15gazebo::rendering::Camera Class Reference . . . 186
10.15.1 Detailed Description . . . 192
10.15.2 Constructor & Destructor Documentation . . . 192
10.15.2.1 Camera . . . 192
10.15.2.2 ∼Camera . . . 193
10.15.3 Member Function Documentation . . . 193
10.15.3.1 AnimationComplete . . . 193
10.15.3.2 AttachToVisual . . . 193
10.15.3.3 AttachToVisual . . . 193
10.15.3.4 AttachToVisualImpl . . . 193
10.15.3.5 AttachToVisualImpl . . . 194
10.15.3.6 AttachToVisualImpl . . . 194
10.15.3.7 ConnectNewImageFrame . . . 194
10.15.3.8 CreateRenderTexture . . . 195
10.15.3.9 DisconnectNewImageFrame . . . 195
10.15.3.10 EnableSaveFrame . . . 195
10.15.3.11 Fini . . . 195
10.15.3.12 GetAspectRatio . . . 195
10.15.3.13 GetAvgFPS . . . 196
10.15.3.14 GetCameraToViewportRay . . . 196
10.15.3.15 GetCaptureData . . . 196
10.15.3.16 GetDirection . . . 196
10.15.3.17 GetFarClip . . . 196
10.15.3.18 GetFrameFilename . . . 196
10.15.3.19 GetHFOV . . . 197
10.15.3.20 GetImageByteSize . . . 197
10.15.3.21 GetImageByteSize . . . 197
10.15.3.22 GetImageData . . . 197
10.15.3.23 GetImageDepth . . . 197
10.15.3.24 GetImageFormat . . . 198

10.15.3.25GetImageHeight . . . 198
10.15.3.26GetImageWidth . . . 198
10.15.3.27GetInitialized . . . 198
10.15.3.28GetLastRenderWallTime . . . 198
10.15.3.29GetName . . . 199
10.15.3.30GetNearClip . . . 199
10.15.3.31GetOgreCamera . . . 199
10.15.3.32GetPitchNode . . . 199
10.15.3.33GetRenderRate . . . 199
10.15.3.34GetRenderTexture . . . 199
10.15.3.35GetRight . . . 200
10.15.3.36GetScene . . . 200
10.15.3.37GetSceneNode . . . 200
10.15.3.38GetScreenshotPath . . . 200
10.15.3.39GetTextureHeight . . . 200
10.15.3.40GetTextureWidth . . . 200
10.15.3.41GetTriangleCount . . . 201
10.15.3.42GetUp . . . 201
10.15.3.43GetVFOV . . . 201
10.15.3.44GetViewport . . . 201
10.15.3.45GetViewportHeight . . . 201
10.15.3.46GetViewportWidth . . . 201
10.15.3.47GetWindowId . . . 202
10.15.3.48GetWorldPointOnPlane . . . 202
10.15.3.49GetWorldPose . . . 202
10.15.3.50GetWorldPosition . . . 202
10.15.3.51GetWorldRotation . . . 202
10.15.3.52GetZValue . . . 203
10.15.3.53Init . . . 203
10.15.3.54IsAnimating . . . 203
10.15.3.55IsVisible . . . 203
10.15.3.56IsVisible . . . 203
10.15.3.57Load . . . 204
10.15.3.58Load . . . 204
10.15.3.59MoveToPosition . . . 204
10.15.3.60MoveToPositions . . . 204
10.15.3.61PostRender . . . 204

10.15.3.62 ReadPixelBuffer . . . 205
10.15.3.63 Render . . . 205
10.15.3.64 Render . . . 205
10.15.3.65 RenderImpl . . . 205
10.15.3.66 RotatePitch . . . 205
10.15.3.67 RotateYaw . . . 205
10.15.3.68 SaveFrame . . . 205
10.15.3.69 SaveFrame . . . 206
10.15.3.70 SetAspectRatio . . . 206
10.15.3.71 SetCaptureData . . . 206
10.15.3.72 SetCaptureDataOnce . . . 206
10.15.3.73 SetClipDist . . . 206
10.15.3.74 SetHFOV . . . 207
10.15.3.75 SetImageHeight . . . 207
10.15.3.76 SetImageSize . . . 207
10.15.3.77 SetImageWidth . . . 207
10.15.3.78 SetName . . . 207
10.15.3.79 SetRenderRate . . . 208
10.15.3.80 SetRenderTarget . . . 208
10.15.3.81 SetSaveFramePathname . . . 208
10.15.3.82 SetScene . . . 208
10.15.3.83 SetSceneNode . . . 208
10.15.3.84 SetWindowId . . . 208
10.15.3.85 SetWorldPose . . . 208
10.15.3.86 SetWorldPosition . . . 209
10.15.3.87 SetWorldRotation . . . 209
10.15.3.88 ShowWireframe . . . 209
10.15.3.89 ToggleShowWireframe . . . 209
10.15.3.90 TrackVisual . . . 209
10.15.3.91 TrackVisualImpl . . . 209
10.15.3.92 TrackVisualImpl . . . 210
10.15.3.93 Translate . . . 210
10.15.3.94 Update . . . 210
10.15.4 Member Data Documentation . . . 210
10.15.4.1 animState . . . 210
10.15.4.2 bayerFrameBuffer . . . 210
10.15.4.3 camera . . . 210

10.15.4.4 captureData . . . 210
10.15.4.5 captureDataOnce . . . 211
10.15.4.6 connections . . . 211
10.15.4.7 imageFormat . . . 211
10.15.4.8 imageHeight . . . 211
10.15.4.9 imageWidth . . . 211
10.15.4.10 initialized . . . 211
10.15.4.11 lastRenderWallTime . . . 211
10.15.4.12 name . . . 211
10.15.4.13 newData . . . 211
10.15.4.14 newImageFrame . . . 211
10.15.4.15 onAnimationComplete . . . 211
10.15.4.16 pitchNode . . . 212
10.15.4.17 prevAnimTime . . . 212
10.15.4.18 renderTarget . . . 212
10.15.4.19 renderTexture . . . 212
10.15.4.20 requests . . . 212
10.15.4.21 saveCount . . . 212
10.15.4.22 saveFrameBuffer . . . 212
10.15.4.23 scene . . . 212
10.15.4.24 sceneNode . . . 212
10.15.4.25 screenshotPath . . . 212
10.15.4.26 sdf . . . 212
10.15.4.27 textureHeight . . . 212
10.15.4.28 textureWidth . . . 213
10.15.4.29 viewport . . . 213
10.15.4.30 windowId . . . 213
10.16 gazebo::sensors::CameraSensor Class Reference . . . 213
10.16.1 Detailed Description . . . 214
10.16.2 Constructor & Destructor Documentation . . . 214
10.16.2.1 CameraSensor . . . 214
10.16.2.2 ∼CameraSensor . . . 214
10.16.3 Member Function Documentation . . . 215
10.16.3.1 Fini . . . 215
10.16.3.2 GetCamera . . . 215
10.16.3.3 GetImageData . . . 215
10.16.3.4 GetImageHeight . . . 215

10.16.3.5 GetImageWidth . . . 215
10.16.3.6 GetTopic . . . 215
10.16.3.7 Init . . . 216
10.16.3.8 IsActive . . . 216
10.16.3.9 Load . . . 216
10.16.3.10 Load . . . 216
10.16.3.11 SaveFrame . . . 216
10.16.3.12 UpdateImpl . . . 217
10.17 gazebo::rendering::CameraVisual Class Reference . . . 217
10.17.1 Detailed Description . . . 218
10.17.2 Constructor & Destructor Documentation . . . 218
10.17.2.1 CameraVisual . . . 218
10.17.2.2 ∼CameraVisual . . . 218
10.17.3 Member Function Documentation . . . 218
10.17.3.1 Load . . . 218
10.18 gazebo::common::ColladaLoader Class Reference . . . 218
10.18.1 Detailed Description . . . 219
10.18.2 Constructor & Destructor Documentation . . . 219
10.18.2.1 ColladaLoader . . . 219
10.18.2.2 ∼ColladaLoader . . . 219
10.18.3 Member Function Documentation . . . 219
10.18.3.1 Load . . . 219
10.19 gazebo::physics::Collision Class Reference . . . 220
10.19.1 Detailed Description . . . 222
10.19.2 Constructor & Destructor Documentation . . . 222
10.19.2.1 Collision . . . 222
10.19.2.2 ∼Collision . . . 223
10.19.3 Member Function Documentation . . . 223
10.19.3.1 AddContact . . . 223
10.19.3.2 FillMsg . . . 223
10.19.3.3 Fini . . . 223
10.19.3.4 GetBoundingBox . . . 223
10.19.3.5 GetContactsEnabled . . . 223
10.19.3.6 GetLaserRetro . . . 224
10.19.3.7 GetLink . . . 224
10.19.3.8 GetMaxContacts . . . 224
10.19.3.9 GetModel . . . 224

10.19.3.10GetRelativeAngularAccel . . . 224
10.19.3.11GetRelativeAngularVel . . . 224
10.19.3.12GetRelativeLinearAccel . . . 225
10.19.3.13GetRelativeLinearVel . . . 225
10.19.3.14GetShape . . . 225
10.19.3.15GetShapeType . . . 225
10.19.3.16GetState . . . 225
10.19.3.17GetSurface . . . 226
10.19.3.18GetWorldAngularAccel . . . 226
10.19.3.19GetWorldAngularVel . . . 226
10.19.3.20GetWorldLinearAccel . . . 226
10.19.3.21GetWorldLinearVel . . . 226
10.19.3.22Init . . . 226
10.19.3.23IsPlaceable . . . 227
10.19.3.24Load . . . 227
10.19.3.25ProcessMsg . . . 227
10.19.3.26SetCategoryBits . . . 227
10.19.3.27SetCollideBits . . . 227
10.19.3.28SetCollision . . . 228
10.19.3.29SetContactsEnabled . . . 228
10.19.3.30SetLaserRetro . . . 228
10.19.3.31SetMaxContacts . . . 228
10.19.3.32SetScale . . . 228
10.19.3.33SetShape . . . 228
10.19.3.34SetState . . . 229
10.19.3.35UpdateParameters . . . 229
10.19.4 Member Data Documentation . . . 229
10.19.4.1 link . . . 229
10.19.4.2 placeable . . . 229
10.19.4.3 shape . . . 229
10.20gazebo::physics::CollisionState Class Reference . . . 229
10.20.1 Detailed Description . . . 231
10.20.2 Constructor & Destructor Documentation . . . 231
10.20.2.1 CollisionState . . . 231
10.20.2.2 CollisionState . . . 231
10.20.2.3 CollisionState . . . 231
10.20.2.4 ∼CollisionState . . . 231

10.20.3 Member Function Documentation . . . 231
10.20.3.1 FillSDF . . . 231
10.20.3.2 GetPose . . . 232
10.20.3.3 IsZero . . . 232
10.20.3.4 Load . . . 232
10.20.3.5 operator+ . . . 232
10.20.3.6 operator- . . . 232
10.20.3.7 operator= . . . 233
10.20.4 Friends And Related Function Documentation . . . 233
10.20.4.1 operator<< . . . 233
10.21 gazebo::common::Color Class Reference . . . 233
10.21.1 Detailed Description . . . 236
10.21.2 Member Typedef Documentation . . . 236
10.21.2.1 ABGR . . . 236
10.21.2.2 ARGB . . . 236
10.21.2.3 BGRA . . . 236
10.21.2.4 RGBA . . . 236
10.21.3 Constructor & Destructor Documentation . . . 236
10.21.3.1 Color . . . 236
10.21.3.2 Color . . . 236
10.21.3.3 Color . . . 236
10.21.3.4 ~Color . . . 237
10.21.4 Member Function Documentation . . . 237
10.21.4.1 GetAsABGR . . . 237
10.21.4.2 GetAsARGB . . . 237
10.21.4.3 GetAsBGRA . . . 237
10.21.4.4 GetAsHSV . . . 237
10.21.4.5 GetAsRGBA . . . 237
10.21.4.6 GetAsYUV . . . 238
10.21.4.7 operator!= . . . 238
10.21.4.8 operator* . . . 238
10.21.4.9 operator* . . . 238
10.21.4.10 operator*= . . . 238
10.21.4.11 operator+ . . . 239
10.21.4.12 operator+ . . . 239
10.21.4.13 operator+= . . . 239
10.21.4.14 operator- . . . 239

10.21.4.15operator- . . . 240
10.21.4.16operator-= . . . 240
10.21.4.17operator/ . . . 240
10.21.4.18operator/ . . . 240
10.21.4.19operator/= . . . 241
10.21.4.20operator= . . . 241
10.21.4.21operator== . . . 241
10.21.4.22operator[] . . . 241
10.21.4.23Reset . . . 242
10.21.4.24Set . . . 242
10.21.4.25SetFromABGR . . . 242
10.21.4.26SetFromARGB . . . 242
10.21.4.27SetFromBGRA . . . 242
10.21.4.28SetFromHSV . . . 243
10.21.4.29SetFromRGBA . . . 243
10.21.4.30SetFromYUV . . . 243
10.21.5 Friends And Related Function Documentation . . . 243
10.21.5.1 operator<< . . . 243
10.21.5.2 operator>> . . . 243
10.21.6 Member Data Documentation . . . 244
10.21.6.1 a . . . 244
10.21.6.2 b . . . 244
10.21.6.3 Black . . . 244
10.21.6.4 Blue . . . 244
10.21.6.5 g . . . 244
10.21.6.6 Green . . . 244
10.21.6.7 Purple . . . 244
10.21.6.8 r . . . 244
10.21.6.9 Red . . . 244
10.21.6.10White . . . 244
10.21.6.11Yellow . . . 244
10.22gazebo::rendering::COMVisual Class Reference . . . 244
10.22.1 Detailed Description . . . 245
10.22.2 Constructor & Destructor Documentation . . . 245
10.22.2.1 COMVisual . . . 245
10.22.2.2 ∼COMVisual . . . 246
10.22.3 Member Function Documentation . . . 246

10.22.3.1 Load . . . 246
10.22.3.2 Load . . . 246
10.23gazebo::event::Connection Class Reference . . . 246
10.23.1 Detailed Description . . . 247
10.23.2 Constructor & Destructor Documentation . . . 247
10.23.2.1 Connection . . . 247
10.23.2.2 Connection . . . 247
10.23.2.3 ∼Connection . . . 247
10.23.3 Member Function Documentation . . . 247
10.23.3.1 GetId . . . 247
10.24gazebo::transport::Connection Class Reference . . . 247
10.24.1 Detailed Description . . . 249
10.24.2 Member Typedef Documentation . . . 250
10.24.2.1 AcceptCallback . . . 250
10.24.2.2 ReadCallback . . . 250
10.24.3 Constructor & Destructor Documentation . . . 250
10.24.3.1 Connection . . . 250
10.24.3.2 ∼Connection . . . 250
10.24.4 Member Function Documentation . . . 250
10.24.4.1 AsyncRead . . . 250
10.24.4.2 Cancel . . . 250
10.24.4.3 Connect . . . 250
10.24.4.4 ConnectToShutdown . . . 250
10.24.4.5 DisconnectShutdown . . . 251
10.24.4.6 EnqueueMsg . . . 251
10.24.4.7 EnqueueMsg . . . 251
10.24.4.8 GetId . . . 251
10.24.4.9 GetIPWhiteList . . . 252
10.24.4.10 GetLocalAddress . . . 252
10.24.4.11 GetLocalHostname . . . 252
10.24.4.12 GetLocalPort . . . 252
10.24.4.13 GetLocalURI . . . 252
10.24.4.14 GetRemoteAddress . . . 252
10.24.4.15 GetRemoteHostname . . . 253
10.24.4.16 GetRemotePort . . . 253
10.24.4.17 GetRemoteURI . . . 253
10.24.4.18 IsOpen . . . 253

10.24.4.19Listen . . . 253
10.24.4.20ProcessWriteQueue . . . 253
10.24.4.21Read . . . 254
10.24.4.22Shutdown . . . 254
10.24.4.23StartRead . . . 254
10.24.4.24StopRead . . . 254
10.24.4.25ValidateIP . . . 254
10.25gazebo::transport::ConnectionManager Class Reference . . . 254
10.25.1 Detailed Description . . . 256
10.25.2 Member Function Documentation . . . 256
10.25.2.1 Advertise . . . 256
10.25.2.2 ConnectToRemoteHost . . . 256
10.25.2.3 Fini . . . 256
10.25.2.4 GetAllPublishers . . . 256
10.25.2.5 GetTopicNamespaces . . . 257
10.25.2.6 Init . . . 257
10.25.2.7 IsRunning . . . 257
10.25.2.8 RegisterTopicNamespace . . . 257
10.25.2.9 RemoveConnection . . . 257
10.25.2.10Run . . . 258
10.25.2.11Stop . . . 258
10.25.2.12Subscribe . . . 258
10.25.2.13TriggerUpdate . . . 258
10.25.2.14Unadvertise . . . 258
10.25.2.15Unsubscribe . . . 258
10.25.2.16Unsubscribe . . . 258
10.25.3 Member Data Documentation . . . 259
10.25.3.1 eventConnections . . . 259
10.26gazebo::common::Console Class Reference . . . 259
10.26.1 Detailed Description . . . 260
10.27gazebo::physics::Contact Class Reference . . . 260
10.27.1 Detailed Description . . . 261
10.27.2 Constructor & Destructor Documentation . . . 261
10.27.2.1 Contact . . . 261
10.27.2.2 Contact . . . 261
10.27.2.3 ∼Contact . . . 261
10.27.3 Member Function Documentation . . . 261

10.27.3.1 DebugString . . . 261
10.27.3.2 FillMsg . . . 261
10.27.3.3 operator= . . . 262
10.27.3.4 operator= . . . 262
10.27.3.5 Reset . . . 262
10.27.4 Member Data Documentation . . . 262
10.27.4.1 collision1 . . . 262
10.27.4.2 collision2 . . . 262
10.27.4.3 count . . . 262
10.27.4.4 depths . . . 263
10.27.4.5 normals . . . 263
10.27.4.6 positions . . . 263
10.27.4.7 time . . . 263
10.27.4.8 world . . . 263
10.27.4.9 wrench . . . 263
10.28gazebo::physics::ContactManager Class Reference . . . 263
10.28.1 Detailed Description . . . 264
10.28.2 Constructor & Destructor Documentation . . . 264
10.28.2.1 ContactManager . . . 264
10.28.2.2 ∼ContactManager . . . 264
10.28.3 Member Function Documentation . . . 264
10.28.3.1 Clear . . . 264
10.28.3.2 CreateFilter . . . 264
10.28.3.3 CreateFilter . . . 265
10.28.3.4 CreateFilter . . . 265
10.28.3.5 GetContact . . . 265
10.28.3.6 GetContactCount . . . 265
10.28.3.7 GetContacts . . . 265
10.28.3.8 Init . . . 266
10.28.3.9 NewContact . . . 266
10.28.3.10 PublishContacts . . . 266
10.28.3.11 ResetCount . . . 266
10.29gazebo::physics::ContactPublisher Class Reference . . . 266
10.29.1 Detailed Description . . . 267
10.29.2 Member Data Documentation . . . 267
10.29.2.1 collisionNames . . . 267
10.29.2.2 collisions . . . 267

10.29.2.3 contacts . . . 267
10.29.2.4 publisher . . . 267
10.30 gazebo::sensors::ContactSensor Class Reference . . . 267
10.30.1 Detailed Description . . . 269
10.30.2 Constructor & Destructor Documentation . . . 269
10.30.2.1 ContactSensor . . . 269
10.30.2.2 ∼ContactSensor . . . 269
10.30.3 Member Function Documentation . . . 269
10.30.3.1 Fini . . . 269
10.30.3.2 GetCollisionContactCount . . . 269
10.30.3.3 GetCollisionCount . . . 269
10.30.3.4 GetCollisionName . . . 270
10.30.3.5 GetContacts . . . 270
10.30.3.6 GetContacts . . . 271
10.30.3.7 Init . . . 271
10.30.3.8 IsActive . . . 271
10.30.3.9 Load . . . 271
10.30.3.10 Load . . . 271
10.30.3.11 UpdateImpl . . . 272
10.31 gazebo::rendering::ContactVisual Class Reference . . . 272
10.31.1 Detailed Description . . . 273
10.31.2 Constructor & Destructor Documentation . . . 273
10.31.2.1 ContactVisual . . . 273
10.31.2.2 ∼ContactVisual . . . 273
10.31.3 Member Function Documentation . . . 273
10.31.3.1 SetEnabled . . . 273
10.32 gazebo::rendering::Conversions Class Reference . . . 273
10.32.1 Detailed Description . . . 274
10.32.2 Member Function Documentation . . . 274
10.32.2.1 Convert . . . 274
10.32.2.2 Convert . . . 274
10.32.2.3 Convert . . . 274
10.32.2.4 Convert . . . 275
10.32.2.5 Convert . . . 275
10.32.2.6 Convert . . . 275
10.33 gazebo::physics::CylinderShape Class Reference . . . 275
10.33.1 Detailed Description . . . 277

10.33.2 Constructor & Destructor Documentation . . . 277
10.33.2.1 CylinderShape . . . 277
10.33.2.2 ∼CylinderShape . . . 277
10.33.3 Member Function Documentation . . . 277
10.33.3.1 FillMsg . . . 277
10.33.3.2 GetLength . . . 277
10.33.3.3 GetRadius . . . 278
10.33.3.4 Init . . . 278
10.33.3.5 ProcessMsg . . . 278
10.33.3.6 SetLength . . . 278
10.33.3.7 SetRadius . . . 278
10.33.3.8 SetScale . . . 278
10.33.3.9 SetSize . . . 279
10.34 gazebo::physics::DARTBallJoint Class Reference . . . 279
10.34.1 Detailed Description . . . 281
10.34.2 Constructor & Destructor Documentation . . . 281
10.34.2.1 DARTBallJoint . . . 281
10.34.2.2 ∼DARTBallJoint . . . 281
10.34.3 Member Function Documentation . . . 281
10.34.3.1 GetAnchor . . . 282
10.34.3.2 GetAngleImpl . . . 282
10.34.3.3 GetGlobalAxis . . . 282
10.34.3.4 GetMaxForce . . . 282
10.34.3.5 GetVelocity . . . 283
10.34.3.6 Init . . . 283
10.34.3.7 Load . . . 283
10.34.3.8 SetForceImpl . . . 283
10.34.3.9 SetMaxForce . . . 284
10.34.3.10 SetVelocity . . . 284
10.34.4 Member Data Documentation . . . 284
10.34.4.1 dtBallJoint . . . 284
10.35 gazebo::physics::DARTBoxShape Class Reference . . . 284
10.35.1 Detailed Description . . . 285
10.35.2 Constructor & Destructor Documentation . . . 285
10.35.2.1 DARTBoxShape . . . 286
10.35.2.2 ∼DARTBoxShape . . . 286
10.35.3 Member Function Documentation . . . 286

10.35.3.1 SetSize . . . 286
10.36 gazebo::physics::DARTCollision Class Reference . . . 286
10.36.1 Detailed Description . . . 288
10.36.2 Constructor & Destructor Documentation . . . 288
10.36.2.1 DARTCollision . . . 288
10.36.2.2 ~DARTCollision . . . 288
10.36.3 Member Function Documentation . . . 288
10.36.3.1 Fini . . . 288
10.36.3.2 GetBoundingBox . . . 288
10.36.3.3 GetCategoryBits . . . 289
10.36.3.4 GetCollideBits . . . 289
10.36.3.5 GetDARTBodyNode . . . 289
10.36.3.6 GetDARTCollisionShape . . . 289
10.36.3.7 Init . . . 289
10.36.3.8 Load . . . 289
10.36.3.9 OnPoseChange . . . 290
10.36.3.10 SetCategoryBits . . . 290
10.36.3.11 SetCollideBits . . . 290
10.36.3.12 SetDARTCollisionShape . . . 290
10.37 gazebo::physics::DARTCylinderShape Class Reference . . . 290
10.37.1 Detailed Description . . . 291
10.37.2 Constructor & Destructor Documentation . . . 292
10.37.2.1 DARTCylinderShape . . . 292
10.37.2.2 ~DARTCylinderShape . . . 292
10.37.3 Member Function Documentation . . . 292
10.37.3.1 SetSize . . . 292
10.38 gazebo::physics::DARTHeightmapShape Class Reference . . . 292
10.38.1 Detailed Description . . . 293
10.38.2 Constructor & Destructor Documentation . . . 294
10.38.2.1 DARTHeightmapShape . . . 294
10.38.2.2 ~DARTHeightmapShape . . . 294
10.38.3 Member Function Documentation . . . 294
10.38.3.1 Init . . . 294
10.39 gazebo::physics::DARTHinge2Joint Class Reference . . . 294
10.39.1 Detailed Description . . . 296
10.39.2 Constructor & Destructor Documentation . . . 296
10.39.2.1 DARTHinge2Joint . . . 296

10.39.2.2 ~DARTHinge2Joint . . . 296
10.39.3 Member Function Documentation . . . 297
10.39.3.1 GetAnchor . . . 297
10.39.3.2 GetAngleImpl . . . 297
10.39.3.3 GetGlobalAxis . . . 297
10.39.3.4 GetMaxForce . . . 297
10.39.3.5 GetVelocity . . . 298
10.39.3.6 Init . . . 298
10.39.3.7 Load . . . 298
10.39.3.8 SetAxis . . . 298
10.39.3.9 SetForceImpl . . . 298
10.39.3.10 SetMaxForce . . . 299
10.39.3.11 SetVelocity . . . 299
10.39.4 Member Data Documentation . . . 299
10.39.4.1 dtUniveralJoint . . . 299
10.40 gazebo::physics::DARTHingeJoint Class Reference . . . 299
10.40.1 Detailed Description . . . 301
10.40.2 Constructor & Destructor Documentation . . . 301
10.40.2.1 DARTHingeJoint . . . 301
10.40.2.2 ~DARTHingeJoint . . . 301
10.40.3 Member Function Documentation . . . 302
10.40.3.1 GetAnchor . . . 302
10.40.3.2 GetAngleImpl . . . 302
10.40.3.3 GetGlobalAxis . . . 302
10.40.3.4 GetMaxForce . . . 302
10.40.3.5 GetVelocity . . . 303
10.40.3.6 Init . . . 303
10.40.3.7 Load . . . 303
10.40.3.8 SetAxis . . . 303
10.40.3.9 SetForceImpl . . . 303
10.40.3.10 SetMaxForce . . . 304
10.40.3.11 SetVelocity . . . 304
10.40.4 Member Data Documentation . . . 304
10.40.4.1 dtRevoluteJoint . . . 304
10.41 gazebo::physics::DARTJoint Class Reference . . . 304
10.41.1 Detailed Description . . . 306
10.41.2 Constructor & Destructor Documentation . . . 307

10.41.2.1 DARTJoint . . . 307
10.41.2.2 ∼DARTJoint . . . 307
10.41.3 Member Function Documentation . . . 307
10.41.3.1 ApplyDamping . . . 307
10.41.3.2 AreConnected . . . 307
10.41.3.3 Attach . . . 307
10.41.3.4 Detach . . . 308
10.41.3.5 GetAngleCount . . . 308
10.41.3.6 GetAttribute . . . 308
10.41.3.7 GetDARTJoint . . . 308
10.41.3.8 GetDARTModel . . . 308
10.41.3.9 GetForce . . . 308
10.41.3.10 GetForceTorque . . . 309
10.41.3.11 GetForceTorque . . . 309
10.41.3.12 GetHighStop . . . 309
10.41.3.13 GetJointLink . . . 310
10.41.3.14 GetLinkForce . . . 310
10.41.3.15 GetLinkTorque . . . 310
10.41.3.16 GetLowStop . . . 310
10.41.3.17 Init . . . 311
10.41.3.18 Load . . . 311
10.41.3.19 Reset . . . 311
10.41.3.20 SetAnchor . . . 311
10.41.3.21 SetAttribute . . . 312
10.41.3.22 SetDamping . . . 312
10.41.3.23 SetForce . . . 312
10.41.3.24 SetForceImpl . . . 312
10.41.3.25 SetHighStop . . . 313
10.41.3.26 SetLowStop . . . 313
10.41.4 Member Data Documentation . . . 313
10.41.4.1 dartPhysicsEngine . . . 313
10.41.4.2 dtChildBodyNode . . . 313
10.41.4.3 dtJoint . . . 313
10.42 gazebo::physics::DARTLink Class Reference . . . 313
10.42.1 Detailed Description . . . 316
10.42.2 Constructor & Destructor Documentation . . . 316
10.42.2.1 DARTLink . . . 316

10.42.2.2 ~DARTLink . . . 316
10.42.3 Member Function Documentation . . . 316
10.42.3.1 AddDARTChildJoint . . . 316
10.42.3.2 AddForce . . . 317
10.42.3.3 AddForceAtRelativePosition . . . 317
10.42.3.4 AddForceAtWorldPosition . . . 317
10.42.3.5 AddRelativeForce . . . 317
10.42.3.6 AddRelativeTorque . . . 317
10.42.3.7 AddTorque . . . 318
10.42.3.8 Fini . . . 318
10.42.3.9 GetDARTBodyNode . . . 318
10.42.3.10 GetDARTModel . . . 318
10.42.3.11 GetDARTPhysics . . . 318
10.42.3.12 GetDARTWorld . . . 318
10.42.3.13 GetEnabled . . . 319
10.42.3.14 GetGravityMode . . . 319
10.42.3.15 GetKinematic . . . 319
10.42.3.16 GetWorldAngularVel . . . 319
10.42.3.17 GetWorldCoGLinearVel . . . 319
10.42.3.18 GetWorldForce . . . 320
10.42.3.19 GetWorldLinearVel . . . 320
10.42.3.20 GetWorldLinearVel . . . 320
10.42.3.21 GetWorldTorque . . . 320
10.42.3.22 Init . . . 321
10.42.3.23 Load . . . 321
10.42.3.24 OnPoseChange . . . 321
10.42.3.25 SetAngularDamping . . . 321
10.42.3.26 SetAngularVel . . . 321
10.42.3.27 SetAutoDisable . . . 321
10.42.3.28 SetDARTParentJoint . . . 322
10.42.3.29 SetEnabled . . . 322
10.42.3.30 SetForce . . . 322
10.42.3.31 SetGravityMode . . . 322
10.42.3.32 SetKinematic . . . 322
10.42.3.33 SetLinearDamping . . . 323
10.42.3.34 SetLinearVel . . . 323
10.42.3.35 SetLinkStatic . . . 323

10.42.3.36 SetSelfCollide . . . 323
10.42.3.37 SetTorque . . . 323
10.42.3.38 updateDirtyPoseFromDARTTransformation . . . 324
10.43 gazebo::physics::DARTMeshShape Class Reference . . . 324
10.43.1 Detailed Description . . . 325
10.43.2 Constructor & Destructor Documentation . . . 325
10.43.2.1 DARTMeshShape . . . 325
10.43.2.2 ∼DARTMeshShape . . . 325
10.43.3 Member Function Documentation . . . 325
10.43.3.1 Init . . . 325
10.43.3.2 Load . . . 325
10.43.3.3 Update . . . 326
10.44 gazebo::physics::DARTModel Class Reference . . . 326
10.44.1 Detailed Description . . . 327
10.44.2 Constructor & Destructor Documentation . . . 327
10.44.2.1 DARTModel . . . 327
10.44.2.2 ∼DARTModel . . . 327
10.44.3 Member Function Documentation . . . 327
10.44.3.1 BackupState . . . 327
10.44.3.2 Fini . . . 328
10.44.3.3 GetDARTPhysics . . . 328
10.44.3.4 GetDARTSkeleton . . . 328
10.44.3.5 GetDARTWorld . . . 328
10.44.3.6 Init . . . 328
10.44.3.7 Load . . . 328
10.44.3.8 RestoreState . . . 328
10.44.3.9 Update . . . 328
10.44.4 Member Data Documentation . . . 328
10.44.4.1 dtConfig . . . 328
10.44.4.2 dtSkeleton . . . 328
10.44.4.3 dtVelocity . . . 328
10.45 gazebo::physics::DARTMultiRayShape Class Reference . . . 328
10.45.1 Detailed Description . . . 330
10.46 gazebo::physics::DARTPhysics Class Reference . . . 330
10.46.1 Detailed Description . . . 331
10.46.2 Member Enumeration Documentation . . . 332
10.46.2.1 DARTParam . . . 332

10.46.3 Constructor & Destructor Documentation . . . 332
10.46.3.1 DARTPhysics . . . 332
10.46.3.2 ∼DARTPhysics . . . 332
10.46.4 Member Function Documentation . . . 332
10.46.4.1 CreateCollision . . . 332
10.46.4.2 CreateJoint . . . 332
10.46.4.3 CreateLink . . . 332
10.46.4.4 CreateModel . . . 333
10.46.4.5 CreateShape . . . 333
10.46.4.6 DebugPrint . . . 333
10.46.4.7 Fini . . . 333
10.46.4.8 GetDARTWorld . . . 333
10.46.4.9 GetParam . . . 333
10.46.4.10 GetParam . . . 334
10.46.4.11 GetType . . . 334
10.46.4.12 Init . . . 334
10.46.4.13 InitForThread . . . 334
10.46.4.14 Load . . . 334
10.46.4.15 OnPhysicsMsg . . . 334
10.46.4.16 OnRequest . . . 335
10.46.4.17 Reset . . . 335
10.46.4.18 SetGravity . . . 335
10.46.4.19 SetSeed . . . 335
10.46.4.20 UpdateCollision . . . 335
10.46.4.21 UpdatePhysics . . . 335
10.47 gazebo::physics::DARTPlaneShape Class Reference . . . 336
10.47.1 Detailed Description . . . 337
10.47.2 Constructor & Destructor Documentation . . . 337
10.47.2.1 DARTPlaneShape . . . 337
10.47.2.2 ∼DARTPlaneShape . . . 337
10.47.3 Member Function Documentation . . . 337
10.47.3.1 CreatePlane . . . 337
10.47.3.2 SetAltitude . . . 337
10.48 gazebo::physics::DARTRayShape Class Reference . . . 338
10.48.1 Detailed Description . . . 339
10.49 gazebo::physics::DARTScrewJoint Class Reference . . . 339
10.49.1 Detailed Description . . . 341

10.49.2 Constructor & Destructor Documentation . . . 341
10.49.2.1 DARTScrewJoint . . . 341
10.49.2.2 ∼DARTScrewJoint . . . 341
10.49.3 Member Function Documentation . . . 342
10.49.3.1 GetAngleImpl . . . 342
10.49.3.2 GetGlobalAxis . . . 342
10.49.3.3 GetMaxForce . . . 342
10.49.3.4 GetThreadPitch . . . 342
10.49.3.5 GetVelocity . . . 343
10.49.3.6 Init . . . 343
10.49.3.7 Load . . . 343
10.49.3.8 SetAxis . . . 343
10.49.3.9 SetForceImpl . . . 344
10.49.3.10 SetMaxForce . . . 344
10.49.3.11 SetThreadPitch . . . 344
10.49.3.12 SetVelocity . . . 344
10.49.4 Member Data Documentation . . . 344
10.49.4.1 dartScrewJoint . . . 345
10.50 gazebo::physics::DARTSliderJoint Class Reference . . . 345
10.50.1 Detailed Description . . . 346
10.50.2 Constructor & Destructor Documentation . . . 347
10.50.2.1 DARTSliderJoint . . . 347
10.50.2.2 ∼DARTSliderJoint . . . 347
10.50.3 Member Function Documentation . . . 347
10.50.3.1 GetAnchor . . . 347
10.50.3.2 GetAngleImpl . . . 347
10.50.3.3 GetGlobalAxis . . . 347
10.50.3.4 GetMaxForce . . . 348
10.50.3.5 GetVelocity . . . 348
10.50.3.6 Init . . . 348
10.50.3.7 Load . . . 348
10.50.3.8 SetAxis . . . 349
10.50.3.9 SetForceImpl . . . 349
10.50.3.10 SetMaxForce . . . 349
10.50.3.11 SetVelocity . . . 349
10.50.4 Member Data Documentation . . . 349
10.50.4.1 dtPrismaticJoint . . . 350

10.51 gazebo::physics::DARTSphereShape Class Reference . . . 350
10.51.1 Detailed Description . . . 351
10.51.2 Constructor & Destructor Documentation . . . 351
10.51.2.1 DARTSphereShape . . . 351
10.51.2.2 ∼DARTSphereShape . . . 351
10.51.3 Member Function Documentation . . . 351
10.51.3.1 SetRadius . . . 351
10.52 gazebo::physics::DARTTypes Class Reference . . . 351
10.52.1 Detailed Description . . . 352
10.53 gazebo::physics::DARTUniversalJoint Class Reference . . . 352
10.53.1 Detailed Description . . . 354
10.53.2 Constructor & Destructor Documentation . . . 354
10.53.2.1 DARTUniversalJoint . . . 354
10.53.2.2 ∼DARTUniversalJoint . . . 354
10.53.3 Member Function Documentation . . . 355
10.53.3.1 GetAnchor . . . 355
10.53.3.2 GetAngleImpl . . . 355
10.53.3.3 GetGlobalAxis . . . 355
10.53.3.4 GetMaxForce . . . 355
10.53.3.5 GetVelocity . . . 356
10.53.3.6 Init . . . 356
10.53.3.7 Load . . . 356
10.53.3.8 SetAxis . . . 356
10.53.3.9 SetForceImpl . . . 356
10.53.3.10 SetMaxForce . . . 357
10.53.3.11 SetVelocity . . . 357
10.53.4 Member Data Documentation . . . 357
10.53.4.1 dtUniveralJoint . . . 357
10.54 gazebo::rendering::DepthCamera Class Reference . . . 357
10.54.1 Detailed Description . . . 359
10.54.2 Constructor & Destructor Documentation . . . 359
10.54.2.1 DepthCamera . . . 359
10.54.2.2 ∼DepthCamera . . . 359
10.54.3 Member Function Documentation . . . 359
10.54.3.1 ConnectNewDepthFrame . . . 359
10.54.3.2 ConnectNewRGBPointCloud . . . 360
10.54.3.3 CreateDepthTexture . . . 360

10.54.3.4 DisconnectNewDepthFrame . . . 360
10.54.3.5 DisconnectNewRGBPointCloud . . . 360
10.54.3.6 Fini . . . 361
10.54.3.7 GetDepthData . . . 361
10.54.3.8 Init . . . 361
10.54.3.9 Load . . . 361
10.54.3.10 Load . . . 361
10.54.3.11 PostRender . . . 361
10.54.3.12 SetDepthTarget . . . 361
10.54.4 Member Data Documentation . . . 362
10.54.4.1 depthTarget . . . 362
10.54.4.2 depthTexture . . . 362
10.54.4.3 depthViewport . . . 362
10.55 gazebo::sensors::DepthCameraSensor Class Reference . . . 362
10.55.1 Constructor & Destructor Documentation . . . 363
10.55.1.1 DepthCameraSensor . . . 363
10.55.1.2 ~DepthCameraSensor . . . 363
10.55.2 Member Function Documentation . . . 363
10.55.2.1 Fini . . . 363
10.55.2.2 GetDepthCamera . . . 363
10.55.2.3 Init . . . 364
10.55.2.4 Load . . . 364
10.55.2.5 Load . . . 364
10.55.2.6 SaveFrame . . . 364
10.55.2.7 SetActive . . . 364
10.55.2.8 UpdateImpl . . . 365
10.56 gazebo::util::DiagnosticManager Class Reference . . . 365
10.56.1 Detailed Description . . . 366
10.56.2 Member Function Documentation . . . 366
10.56.2.1 GetLabel . . . 366
10.56.2.2 GetLogPath . . . 366
10.56.2.3 GetTime . . . 366
10.56.2.4 GetTime . . . 367
10.56.2.5 GetTimerCount . . . 367
10.56.2.6 Init . . . 367
10.56.2.7 Lap . . . 367
10.56.2.8 StartTimer . . . 367

10.56.2.9 StopTimer . . . 368
10.57gazebo::util::DiagnosticTimer Class Reference . . . 368
10.57.1 Detailed Description . . . 369
10.57.2 Constructor & Destructor Documentation . . . 369
10.57.2.1 DiagnosticTimer . . . 369
10.57.2.2 ∼DiagnosticTimer . . . 369
10.57.3 Member Function Documentation . . . 369
10.57.3.1 GetName . . . 369
10.57.3.2 Lap . . . 369
10.57.3.3 Start . . . 369
10.57.3.4 Stop . . . 369
10.58gazebo::rendering::DummyPageProvider Class Reference . . . 370
10.58.1 Detailed Description . . . 370
10.58.2 Member Function Documentation . . . 370
10.58.2.1 loadProceduralPage . . . 370
10.58.2.2 prepareProceduralPage . . . 371
10.58.2.3 unloadProceduralPage . . . 371
10.58.2.4 unprepareProceduralPage . . . 371
10.59gazebo::rendering::DynamicLines Class Reference . . . 371
10.59.1 Detailed Description . . . 372
10.59.2 Constructor & Destructor Documentation . . . 372
10.59.2.1 DynamicLines . . . 372
10.59.2.2 ∼DynamicLines . . . 373
10.59.3 Member Function Documentation . . . 373
10.59.3.1 AddPoint . . . 373
10.59.3.2 AddPoint . . . 373
10.59.3.3 Clear . . . 373
10.59.3.4 GetMovableType . . . 373
10.59.3.5 getMovableType . . . 373
10.59.3.6 GetPoint . . . 374
10.59.3.7 GetPointCount . . . 374
10.59.3.8 SetColor . . . 374
10.59.3.9 SetPoint . . . 374
10.59.3.10Update . . . 374
10.60gazebo::rendering::DynamicRenderable Class Reference . . . 375
10.60.1 Detailed Description . . . 376
10.60.2 Constructor & Destructor Documentation . . . 376

10.60.2.1 DynamicRenderable . . . 376
10.60.2.2 ∼DynamicRenderable . . . 376
10.60.3 Member Function Documentation . . . 376
10.60.3.1 CreateVertexDeclaration . . . 376
10.60.3.2 FillHardwareBuffers . . . 376
10.60.3.3 getBoundingRadius . . . 377
10.60.3.4 GetMovableType . . . 377
10.60.3.5 GetOperationType . . . 377
10.60.3.6 getSquaredViewDepth . . . 377
10.60.3.7 Init . . . 377
10.60.3.8 PrepareHardwareBuffers . . . 378
10.60.3.9 SetOperationType . . . 378
10.60.4 Member Data Documentation . . . 378
10.60.4.1 indexBufferCapacity . . . 378
10.60.4.2 vertexBufferCapacity . . . 378
10.61 gazebo::physics::Entity Class Reference . . . 379
10.61.1 Detailed Description . . . 381
10.61.2 Constructor & Destructor Documentation . . . 382
10.61.2.1 Entity . . . 382
10.61.2.2 ∼Entity . . . 382
10.61.3 Member Function Documentation . . . 382
10.61.3.1 Fini . . . 382
10.61.3.2 GetBoundingBox . . . 382
10.61.3.3 GetChildCollision . . . 382
10.61.3.4 GetChildLink . . . 383
10.61.3.5 GetCollisionBoundingBox . . . 383
10.61.3.6 GetDirtyPose . . . 383
10.61.3.7 GetInitialRelativePose . . . 383
10.61.3.8 GetNearestEntityBelow . . . 383
10.61.3.9 GetParentModel . . . 384
10.61.3.10 GetRelativeAngularAccel . . . 384
10.61.3.11 GetRelativeAngularVel . . . 384
10.61.3.12 GetRelativeLinearAccel . . . 384
10.61.3.13 GetRelativeLinearVel . . . 384
10.61.3.14 GetRelativePose . . . 385
10.61.3.15 GetWorldAngularAccel . . . 385
10.61.3.16 GetWorldAngularVel . . . 385

10.61.3.17GetWorldLinearAccel . . . 385
10.61.3.18GetWorldLinearVel . . . 385
10.61.3.19GetWorldPose . . . 386
10.61.3.20IsCanonicalLink . . . 386
10.61.3.21IsStatic . . . 386
10.61.3.22Load . . . 386
10.61.3.23OnPoseChange . . . 386
10.61.3.24PlaceOnEntity . . . 386
10.61.3.25PlaceOnNearestEntityBelow . . . 387
10.61.3.26Reset . . . 387
10.61.3.27SetAnimation . . . 387
10.61.3.28SetAnimation . . . 387
10.61.3.29SetCanonicalLink . . . 387
10.61.3.30SetInitialRelativePose . . . 387
10.61.3.31SetName . . . 388
10.61.3.32SetRelativePose . . . 388
10.61.3.33SetStatic . . . 388
10.61.3.34SetWorldPose . . . 388
10.61.3.35SetWorldTwist . . . 388
10.61.3.36StopAnimation . . . 389
10.61.3.37UpdateParameters . . . 389
10.61.4 Member Data Documentation . . . 389
10.61.4.1 animation . . . 389
10.61.4.2 animationConnection . . . 389
10.61.4.3 animationStartPose . . . 389
10.61.4.4 connections . . . 389
10.61.4.5 dirtyPose . . . 389
10.61.4.6 node . . . 389
10.61.4.7 parentEntity . . . 389
10.61.4.8 prevAnimationTime . . . 390
10.61.4.9 requestPub . . . 390
10.61.4.10scale . . . 390
10.61.4.11visPub . . . 390
10.61.4.12visualMsg . . . 390
10.62gazebo::event::Event Class Reference . . . 390
10.62.1 Detailed Description . . . 392
10.62.2 Constructor & Destructor Documentation . . . 392

10.62.2.1 ∼Event . . . 392
10.62.3 Member Function Documentation . . . 392
10.62.3.1 Disconnect . . . 392
10.62.3.2 Disconnect . . . 392
10.63 gazebo::event::Events Class Reference . . . 393
10.63.1 Detailed Description . . . 395
10.63.2 Member Function Documentation . . . 395
10.63.2.1 ConnectAddEntity . . . 395
10.63.2.2 ConnectCreateEntity . . . 396
10.63.2.3 ConnectDeleteEntity . . . 396
10.63.2.4 ConnectDiagTimerStart . . . 396
10.63.2.5 ConnectDiagTimerStop . . . 397
10.63.2.6 ConnectPause . . . 397
10.63.2.7 ConnectPostRender . . . 397
10.63.2.8 ConnectPreRender . . . 397
10.63.2.9 ConnectRender . . . 398
10.63.2.10 ConnectSetSelectedEntity . . . 398
10.63.2.11 ConnectSigInt . . . 398
10.63.2.12 ConnectStep . . . 399
10.63.2.13 ConnectStop . . . 399
10.63.2.14 ConnectWorldCreated . . . 399
10.63.2.15 ConnectWorldUpdateBegin . . . 399
10.63.2.16 ConnectWorldUpdateEnd . . . 400
10.63.2.17 DisconnectAddEntity . . . 400
10.63.2.18 DisconnectCreateEntity . . . 400
10.63.2.19 DisconnectDeleteEntity . . . 400
10.63.2.20 DisconnectDiagTimerStart . . . 401
10.63.2.21 DisconnectDiagTimerStop . . . 401
10.63.2.22 DisconnectPause . . . 401
10.63.2.23 DisconnectPostRender . . . 401
10.63.2.24 DisconnectPreRender . . . 402
10.63.2.25 DisconnectRender . . . 402
10.63.2.26 DisconnectSetSelectedEntity . . . 402
10.63.2.27 DisconnectSigInt . . . 402
10.63.2.28 DisconnectStep . . . 402
10.63.2.29 DisconnectStop . . . 403
10.63.2.30 DisconnectWorldCreated . . . 403

10.63.2.31 DisconnectWorldUpdateBegin . . . 403
10.63.2.32 DisconnectWorldUpdateEnd . . . 403
10.63.3 Member Data Documentation . . . 403
10.63.3.1 addEntity . . . 403
10.63.3.2 deleteEntity . . . 403
10.63.3.3 diagTimerStart . . . 404
10.63.3.4 diagTimerStop . . . 404
10.63.3.5 entityCreated . . . 404
10.63.3.6 pause . . . 404
10.63.3.7 postRender . . . 404
10.63.3.8 preRender . . . 404
10.63.3.9 render . . . 404
10.63.3.10 setSelectedEntity . . . 404
10.63.3.11 sigInt . . . 404
10.63.3.12 step . . . 405
10.63.3.13 stop . . . 405
10.63.3.14 worldCreated . . . 405
10.63.3.15 worldUpdateBegin . . . 405
10.63.3.16 worldUpdateEnd . . . 405
10.64 gazebo::rendering::Events Class Reference . . . 405
10.64.1 Detailed Description . . . 406
10.64.2 Member Function Documentation . . . 406
10.64.2.1 ConnectCreateScene . . . 406
10.64.2.2 ConnectRemoveScene . . . 406
10.64.2.3 DisconnectCreateScene . . . 406
10.64.2.4 DisconnectRemoveScene . . . 407
10.64.3 Member Data Documentation . . . 407
10.64.3.1 createScene . . . 407
10.64.3.2 removeScene . . . 407
10.65 gazebo::event::EventT< T > Class Template Reference . . . 407
10.65.1 Detailed Description . . . 410
10.65.2 Member Function Documentation . . . 410
10.65.2.1 operator() . . . 410
10.65.2.2 operator() . . . 410
10.65.2.3 operator() . . . 410
10.65.2.4 operator() . . . 410
10.65.2.5 operator() . . . 411

10.65.2.6 operator() . . . 411
10.65.2.7 operator() . . . 411
10.65.2.8 operator() . . . 411
10.65.2.9 operator() . . . 412
10.65.2.10 operator() . . . 412
10.65.2.11 operator() . . . 413
10.65.2.12 Signal . . . 413
10.65.2.13 Signal . . . 413
10.65.2.14 Signal . . . 413
10.65.2.15 Signal . . . 413
10.65.2.16 Signal . . . 414
10.65.2.17 Signal . . . 414
10.65.2.18 Signal . . . 414
10.65.2.19 Signal . . . 415
10.65.2.20 Signal . . . 415
10.65.2.21 Signal . . . 415
10.65.2.22 Signal . . . 416
10.66 gazebo::common::Exception Class Reference . . . 416
10.66.1 Detailed Description . . . 417
10.66.2 Constructor & Destructor Documentation . . . 417
10.66.2.1 Exception . . . 417
10.66.2.2 Exception . . . 417
10.66.2.3 ∼Exception . . . 417
10.66.3 Member Function Documentation . . . 417
10.66.3.1 GetErrorFile . . . 418
10.66.3.2 GetErrorStr . . . 418
10.66.3.3 Print . . . 418
10.66.4 Friends And Related Function Documentation . . . 418
10.66.4.1 operator<< . . . 418
10.67 gazebo::sensors::ForceTorqueSensor Class Reference . . . 418
10.67.1 Detailed Description . . . 420
10.67.2 Constructor & Destructor Documentation . . . 420
10.67.2.1 ForceTorqueSensor . . . 420
10.67.2.2 ∼ForceTorqueSensor . . . 420
10.67.3 Member Function Documentation . . . 420
10.67.3.1 ConnectUpdate . . . 420
10.67.3.2 DisconnectUpdate . . . 420

10.67.3.3 Fini . . . 421
10.67.3.4 GetForce . . . 421
10.67.3.5 GetJoint . . . 421
10.67.3.6 GetTopic . . . 421
10.67.3.7 GetTorque . . . 421
10.67.3.8 Init . . . 421
10.67.3.9 IsActive . . . 422
10.67.3.10 Load . . . 422
10.67.3.11 UpdateImpl . . . 422
10.67.4 Member Data Documentation . . . 422
10.67.4.1 update . . . 422
10.68 gazebo::rendering::FPSViewController Class Reference . . . 422
10.68.1 Detailed Description . . . 423
10.68.2 Constructor & Destructor Documentation . . . 424
10.68.2.1 FPSViewController . . . 424
10.68.2.2 ∼FPSViewController . . . 424
10.68.3 Member Function Documentation . . . 424
10.68.3.1 GetTypeString . . . 424
10.68.3.2 HandleKeyPressEvent . . . 424
10.68.3.3 HandleKeyReleaseEvent . . . 424
10.68.3.4 HandleMouseEvent . . . 424
10.68.3.5 Init . . . 425
10.68.3.6 Update . . . 425
10.69 google::protobuf::compiler::cpp::GazeboGenerator Class Reference . . . 425
10.69.1 Detailed Description . . . 426
10.69.2 Constructor & Destructor Documentation . . . 426
10.69.2.1 GazeboGenerator . . . 426
10.69.2.2 ∼GazeboGenerator . . . 426
10.69.3 Member Function Documentation . . . 426
10.69.3.1 Generate . . . 426
10.70 gazebo::sensors::GpsSensor Class Reference . . . 426
10.70.1 Detailed Description . . . 427
10.70.2 Constructor & Destructor Documentation . . . 427
10.70.2.1 GpsSensor . . . 427
10.70.2.2 ∼GpsSensor . . . 427
10.70.3 Member Function Documentation . . . 427
10.70.3.1 Fini . . . 427

10.70.3.2 GetAltitude . . . 428
10.70.3.3 GetLatitude . . . 428
10.70.3.4 GetLongitude . . . 428
10.70.3.5 Init . . . 428
10.70.3.6 Load . . . 428
10.70.3.7 Load . . . 428
10.70.3.8 UpdateImpl . . . 428
10.71 gazebo::rendering::GpuLaser Class Reference . . . 429
10.71.1 Detailed Description . . . 431
10.71.2 Constructor & Destructor Documentation . . . 432
10.71.2.1 GpuLaser . . . 432
10.71.2.2 ∼GpuLaser . . . 432
10.71.3 Member Function Documentation . . . 432
10.71.3.1 ConnectNewLaserFrame . . . 432
10.71.3.2 CreateLaserTexture . . . 432
10.71.3.3 DisconnectNewLaserFrame . . . 432
10.71.3.4 Fini . . . 433
10.71.3.5 GetCameraCount . . . 433
10.71.3.6 GetCosHorzFOV . . . 433
10.71.3.7 GetCosVertFOV . . . 433
10.71.3.8 GetFarClip . . . 433
10.71.3.9 GetHorzFOV . . . 433
10.71.3.10 GetHorzHalfAngle . . . 434
10.71.3.11 GetLaserData . . . 434
10.71.3.12 GetNearClip . . . 434
10.71.3.13 GetRayCountRatio . . . 434
10.71.3.14 GetVertFOV . . . 434
10.71.3.15 GetVertHalfAngle . . . 434
10.71.3.16 Init . . . 435
10.71.3.17 IsHorizontal . . . 435
10.71.3.18 Load . . . 435
10.71.3.19 Load . . . 435
10.71.3.20 notifyRenderSingleObject . . . 435
10.71.3.21 PostRender . . . 435
10.71.3.22 SetCameraCount . . . 435
10.71.3.23 SetCosHorzFOV . . . 435
10.71.3.24 SetCosVertFOV . . . 436

10.71.3.25SetFarClip . . . 436
10.71.3.26SetHorzFOV . . . 436
10.71.3.27SetHorzHalfAngle . . . 436
10.71.3.28SetIsHorizontal . . . 436
10.71.3.29SetNearClip . . . 436
10.71.3.30SetRangeCount . . . 437
10.71.3.31SetRayCountRatio . . . 437
10.71.3.32SetVertFOV . . . 437
10.71.3.33SetVertHalfAngle . . . 437
10.71.4 Member Data Documentation . . . 437
10.71.4.1 cameraCount . . . 437
10.71.4.2 chfov . . . 437
10.71.4.3 cvfov . . . 437
10.71.4.4 far . . . 438
10.71.4.5 hfov . . . 438
10.71.4.6 horzHalfAngle . . . 438
10.71.4.7 isHorizontal . . . 438
10.71.4.8 near . . . 438
10.71.4.9 rayCountRatio . . . 438
10.71.4.10vertHalfAngle . . . 438
10.71.4.11vfov . . . 438
10.72gazebo::sensors::GpuRaySensor Class Reference . . . 438
10.72.1 Constructor & Destructor Documentation . . . 441
10.72.1.1 GpuRaySensor . . . 441
10.72.1.2 ∼GpuRaySensor . . . 442
10.72.2 Member Function Documentation . . . 442
10.72.2.1 ConnectNewLaserFrame . . . 442
10.72.2.2 DisconnectNewLaserFrame . . . 442
10.72.2.3 Fini . . . 442
10.72.2.4 GetAngleMax . . . 442
10.72.2.5 GetAngleMin . . . 442
10.72.2.6 GetAngleResolution . . . 442
10.72.2.7 GetCameraCount . . . 443
10.72.2.8 GetCosHorzFOV . . . 443
10.72.2.9 GetCosVertFOV . . . 443
10.72.2.10GetFiducial . . . 443
10.72.2.11GetHorzFOV . . . 443

10.72.2.12GetHorzHalfAngle . . . 444
10.72.2.13GetLaserCamera . . . 444
10.72.2.14GetRange . . . 444
10.72.2.15GetRangeCount . . . 444
10.72.2.16GetRangeCountRatio . . . 444
10.72.2.17GetRangeMax . . . 445
10.72.2.18GetRangeMin . . . 445
10.72.2.19GetRangeResolution . . . 445
10.72.2.20GetRanges . . . 445
10.72.2.21GetRayCount . . . 445
10.72.2.22GetRayCountRatio . . . 445
10.72.2.23GetRetro . . . 446
10.72.2.24GetTopic . . . 446
10.72.2.25GetVertFOV . . . 446
10.72.2.26GetVertHalfAngle . . . 446
10.72.2.27GetVerticalAngleMax . . . 446
10.72.2.28GetVerticalAngleMin . . . 447
10.72.2.29GetVerticalRangeCount . . . 447
10.72.2.30GetVerticalRayCount . . . 447
10.72.2.31Init . . . 447
10.72.2.32IsActive . . . 447
10.72.2.33IsHorizontal . . . 447
10.72.2.34Load . . . 448
10.72.2.35Load . . . 448
10.72.2.36SetAngleMax . . . 448
10.72.2.37SetAngleMin . . . 448
10.72.2.38SetVerticalAngleMax . . . 448
10.72.2.39SetVerticalAngleMin . . . 448
10.72.2.40UpdateImpl . . . 449
10.72.3 Member Data Documentation . . . 449
10.72.3.1 cameraElem . . . 449
10.72.3.2 horzElem . . . 449
10.72.3.3 horzRangeCount . . . 449
10.72.3.4 horzRayCount . . . 449
10.72.3.5 rangeCountRatio . . . 449
10.72.3.6 rangeElem . . . 449
10.72.3.7 scanElem . . . 449

10.72.3.8 vertElem . . . 449
10.72.3.9 vertRangeCount . . . 450
10.72.3.10 vertRayCount . . . 450
10.73 gazebo::rendering::Grid Class Reference . . . 450
10.73.1 Detailed Description . . . 451
10.73.2 Constructor & Destructor Documentation . . . 451
10.73.2.1 Grid . . . 451
10.73.2.2 ∼Grid . . . 451
10.73.3 Member Function Documentation . . . 451
10.73.3.1 Enable . . . 451
10.73.3.2 GetCellCount . . . 451
10.73.3.3 GetCellLength . . . 451
10.73.3.4 GetColor . . . 452
10.73.3.5 GetHeight . . . 452
10.73.3.6 GetLineWidth . . . 452
10.73.3.7 GetSceneNode . . . 452
10.73.3.8 Init . . . 452
10.73.3.9 SetCellCount . . . 452
10.73.3.10 SetCellLength . . . 453
10.73.3.11 SetColor . . . 453
10.73.3.12 SetHeight . . . 453
10.73.3.13 SetLineWidth . . . 453
10.73.3.14 SetUserData . . . 453
10.74 gazebo::physics::Gripper Class Reference . . . 453
10.74.1 Detailed Description . . . 454
10.74.2 Constructor & Destructor Documentation . . . 454
10.74.2.1 Gripper . . . 454
10.74.2.2 ∼Gripper . . . 454
10.74.3 Member Function Documentation . . . 454
10.74.3.1 GetName . . . 454
10.74.3.2 Init . . . 455
10.74.3.3 IsAttached . . . 455
10.74.3.4 Load . . . 455
10.74.4 Member Data Documentation . . . 455
10.74.4.1 node . . . 455
10.75 gazebo::rendering::GUIOverlay Class Reference . . . 455
10.75.1 Detailed Description . . . 456

10.75.2 Constructor & Destructor Documentation . . . 456
10.75.2.1 GUIOverlay . . . 456
10.75.2.2 ∼GUIOverlay . . . 456
10.75.3 Member Function Documentation . . . 456
10.75.3.1 AttachCameraToImage . . . 456
10.75.3.2 AttachCameraToImage . . . 457
10.75.3.3 ButtonCallback . . . 457
10.75.3.4 CreateWindow . . . 457
10.75.3.5 HandleKeyPressEvent . . . 458
10.75.3.6 HandleKeyReleaseEvent . . . 458
10.75.3.7 HandleMouseEvent . . . 458
10.75.3.8 Hide . . . 458
10.75.3.9 Init . . . 458
10.75.3.10 IsInitialized . . . 459
10.75.3.11 LoadLayout . . . 459
10.75.3.12 Resize . . . 459
10.75.3.13 Show . . . 459
10.75.3.14 Update . . . 459
10.76 gazebo::rendering::GzTerrainMatGen Class Reference . . . 459
10.76.1 Constructor & Destructor Documentation . . . 460
10.76.1.1 GzTerrainMatGen . . . 460
10.76.1.2 ∼GzTerrainMatGen . . . 460
10.77 gazebo::rendering::Heightmap Class Reference . . . 460
10.77.1 Detailed Description . . . 461
10.77.2 Constructor & Destructor Documentation . . . 462
10.77.2.1 Heightmap . . . 462
10.77.2.2 ∼Heightmap . . . 462
10.77.3 Member Function Documentation . . . 462
10.77.3.1 Flatten . . . 462
10.77.3.2 GetAvgHeight . . . 462
10.77.3.3 GetHeight . . . 462
10.77.3.4 GetImage . . . 463
10.77.3.5 GetMouseHit . . . 463
10.77.3.6 GetOgreTerrain . . . 463
10.77.3.7 GetTerrainSubdivisionCount . . . 463
10.77.3.8 Load . . . 463
10.77.3.9 LoadFromMsg . . . 464

10.77.3.10Lower . . . 464
10.77.3.11Raise . . . 464
10.77.3.12SetWireframe . . . 464
10.77.3.13Smooth . . . 465
10.77.3.14SplitHeights . . . 465
10.77.4 Member Data Documentation . . . 465
10.77.4.1 NumTerrainSubdivisions . . . 465
10.78gazebo::physics::HeightmapShape Class Reference . . . 465
10.78.1 Detailed Description . . . 467
10.78.2 Constructor & Destructor Documentation . . . 467
10.78.2.1 HeightmapShape . . . 467
10.78.2.2 ∼HeightmapShape . . . 468
10.78.3 Member Function Documentation . . . 468
10.78.3.1 FillMsg . . . 468
10.78.3.2 GetHeight . . . 468
10.78.3.3 GetImage . . . 468
10.78.3.4 GetMaxHeight . . . 468
10.78.3.5 GetMinHeight . . . 468
10.78.3.6 GetPos . . . 469
10.78.3.7 GetSize . . . 469
10.78.3.8 GetSubSampling . . . 469
10.78.3.9 GetURI . . . 469
10.78.3.10GetVertexCount . . . 469
10.78.3.11Init . . . 469
10.78.3.12Load . . . 470
10.78.3.13ProcessMsg . . . 470
10.78.3.14SetScale . . . 470
10.78.4 Member Data Documentation . . . 470
10.78.4.1 flipY . . . 470
10.78.4.2 heights . . . 470
10.78.4.3 img . . . 470
10.78.4.4 subSampling . . . 470
10.78.4.5 vertSize . . . 471
10.79gazebo::physics::Hinge2Joint< T > Class Template Reference . . . 471
10.79.1 Detailed Description . . . 471
10.79.2 Constructor & Destructor Documentation . . . 472
10.79.2.1 Hinge2Joint . . . 472

10.79.2.2 ∼Hinge2Joint . . . 472
10.79.3 Member Function Documentation . . . 472
10.79.3.1 GetAngleCount . . . 472
10.79.3.2 Load . . . 472
10.80 gazebo::physics::HingeJoint< T > Class Template Reference . . . 472
10.80.1 Detailed Description . . . 473
10.80.2 Constructor & Destructor Documentation . . . 473
10.80.2.1 HingeJoint . . . 473
10.80.2.2 ∼HingeJoint . . . 474
10.80.3 Member Function Documentation . . . 474
10.80.3.1 GetAngleCount . . . 474
10.80.3.2 Init . . . 474
10.80.3.3 Load . . . 474
10.81 gazebo::common::Image Class Reference . . . 474
10.81.1 Detailed Description . . . 475
10.81.2 Member Enumeration Documentation . . . 476
10.81.2.1 PixelFormat . . . 476
10.81.3 Constructor & Destructor Documentation . . . 476
10.81.3.1 Image . . . 476
10.81.3.2 ∼Image . . . 476
10.81.4 Member Function Documentation . . . 476
10.81.4.1 ConvertPixelFormat . . . 477
10.81.4.2 GetAvgColor . . . 477
10.81.4.3 GetBPP . . . 477
10.81.4.4 GetData . . . 477
10.81.4.5 GetFilename . . . 477
10.81.4.6 GetHeight . . . 478
10.81.4.7 GetMaxColor . . . 478
10.81.4.8 GetPitch . . . 478
10.81.4.9 GetPixel . . . 478
10.81.4.10 GetPixelFormat . . . 478
10.81.4.11 GetRGBData . . . 478
10.81.4.12 GetWidth . . . 479
10.81.4.13 Load . . . 479
10.81.4.14 Rescale . . . 479
10.81.4.15 SavePNG . . . 479
10.81.4.16 SetFromData . . . 479

10.81.4.17 Valid . . . 480
10.82 gazebo::sensors::ImuSensor Class Reference . . . 480
10.82.1 Detailed Description . . . 481
10.82.2 Constructor & Destructor Documentation . . . 481
10.82.2.1 ImuSensor . . . 481
10.82.2.2 ~ImuSensor . . . 481
10.82.3 Member Function Documentation . . . 481
10.82.3.1 Fini . . . 481
10.82.3.2 GetAngularVelocity . . . 482
10.82.3.3 GetImuMessage . . . 482
10.82.3.4 GetLinearAcceleration . . . 482
10.82.3.5 GetOrientation . . . 482
10.82.3.6 Init . . . 482
10.82.3.7 IsActive . . . 482
10.82.3.8 Load . . . 483
10.82.3.9 Load . . . 483
10.82.3.10 SetReferencePose . . . 483
10.82.3.11 UpdateImpl . . . 483
10.83 gazebo::physics::Inertial Class Reference . . . 483
10.83.1 Detailed Description . . . 485
10.83.2 Constructor & Destructor Documentation . . . 486
10.83.2.1 Inertial . . . 486
10.83.2.2 Inertial . . . 486
10.83.2.3 Inertial . . . 486
10.83.2.4 ~Inertial . . . 486
10.83.3 Member Function Documentation . . . 486
10.83.3.1 GetCoG . . . 486
10.83.3.2 GetInertial . . . 486
10.83.3.3 GetIXX . . . 487
10.83.3.4 GetIXY . . . 487
10.83.3.5 GetIXZ . . . 487
10.83.3.6 GetIYY . . . 487
10.83.3.7 GetIYZ . . . 487
10.83.3.8 GetIZZ . . . 487
10.83.3.9 GetMass . . . 488
10.83.3.10 GetMOI . . . 488
10.83.3.11 GetMOI . . . 488

10.83.3.12GetPose . . . 488
10.83.3.13GetPrincipalMoments . . . 488
10.83.3.14GetProductsofInertia . . . 488
10.83.3.15Load . . . 489
10.83.3.16operator+ . . . 489
10.83.3.17operator+= . . . 489
10.83.3.18operator= . . . 489
10.83.3.19ProcessMsg . . . 489
10.83.3.20Reset . . . 490
10.83.3.21Rotate . . . 490
10.83.3.22SetCoG . . . 490
10.83.3.23SetCoG . . . 490
10.83.3.24SetCoG . . . 490
10.83.3.25SetCoG . . . 491
10.83.3.26SetInertiaMatrix . . . 491
10.83.3.27SetIXX . . . 491
10.83.3.28SetIXY . . . 491
10.83.3.29SetIXZ . . . 491
10.83.3.30SetIYY . . . 491
10.83.3.31SetIYZ . . . 492
10.83.3.32SetIZZ . . . 492
10.83.3.33SetMass . . . 492
10.83.3.34SetMOI . . . 492
10.83.3.35UpdateParameters . . . 492
10.83.4 Friends And Related Function Documentation . . . 492
10.83.4.1 operator<< . . . 492
10.84gazebo::common::InternalError Class Reference . . . 493
10.84.1 Detailed Description . . . 494
10.84.2 Constructor & Destructor Documentation . . . 494
10.84.2.1 InternalError . . . 494
10.84.2.2 InternalError . . . 494
10.84.2.3 ∼InternalError . . . 494
10.85gazebo::transport::IOManager Class Reference . . . 494
10.85.1 Detailed Description . . . 495
10.85.2 Constructor & Destructor Documentation . . . 495
10.85.2.1 IOManager . . . 495
10.85.2.2 ∼IOManager . . . 495

10.85.3 Member Function Documentation . . . 495
10.85.3.1 DecCount . . . 495
10.85.3.2 GetCount . . . 495
10.85.3.3 GetIO . . . 495
10.85.3.4 IncCount . . . 495
10.85.3.5 Stop . . . 496
10.86gazebo::physics::Joint Class Reference . . . 496
10.86.1 Detailed Description . . . 500
10.86.2 Member Enumeration Documentation . . . 500
10.86.2.1 Attribute . . . 500
10.86.3 Constructor & Destructor Documentation . . . 500
10.86.3.1 Joint . . . 500
10.86.3.2 ∼Joint . . . 500
10.86.4 Member Function Documentation . . . 501
10.86.4.1 ApplyDamping . . . 501
10.86.4.2 AreConnected . . . 501
10.86.4.3 Attach . . . 501
10.86.4.4 CacheForceTorque . . . 501
10.86.4.5 CheckAndTruncateForce . . . 501
10.86.4.6 ConnectJointUpdate . . . 502
10.86.4.7 Detach . . . 502
10.86.4.8 DisconnectJointUpdate . . . 502
10.86.4.9 FillMsg . . . 502
10.86.4.10 GetAnchor . . . 502
10.86.4.11 GetAngle . . . 503
10.86.4.12 GetAngleCount . . . 503
10.86.4.13 GetAngleImpl . . . 503
10.86.4.14 GetAttribute . . . 504
10.86.4.15 GetChild . . . 504
10.86.4.16 GetDamping . . . 504
10.86.4.17 GetDampingCoefficient . . . 504
10.86.4.18 GetEffortLimit . . . 504
10.86.4.19 GetForce . . . 505
10.86.4.20 GetForceTorque . . . 505
10.86.4.21 GetGlobalAxis . . . 505
10.86.4.22 GetHighStop . . . 506
10.86.4.23 GetInertiaRatio . . . 506

10.86.4.24GetInitialAnchorPose . . . 506
10.86.4.25GetJointLink . . . 507
10.86.4.26GetLinkForce . . . 507
10.86.4.27GetLinkTorque . . . 507
10.86.4.28GetLocalAxis . . . 507
10.86.4.29GetLowerLimit . . . 508
10.86.4.30GetLowStop . . . 508
10.86.4.31GetMaxForce . . . 508
10.86.4.32GetParent . . . 509
10.86.4.33GetUpperLimit . . . 509
10.86.4.34GetVelocity . . . 509
10.86.4.35GetVelocityLimit . . . 510
10.86.4.36Init . . . 510
10.86.4.37Load . . . 510
10.86.4.38Load . . . 510
10.86.4.39Reset . . . 511
10.86.4.40SetAnchor . . . 511
10.86.4.41SetAngle . . . 511
10.86.4.42SetAttribute . . . 511
10.86.4.43SetAxis . . . 512
10.86.4.44SetDamping . . . 512
10.86.4.45SetEffortLimit . . . 512
10.86.4.46SetForce . . . 512
10.86.4.47SetHighStop . . . 513
10.86.4.48SetLowStop . . . 513
10.86.4.49SetMaxForce . . . 513
10.86.4.50SetModel . . . 514
10.86.4.51SetProvideFeedback . . . 514
10.86.4.52SetState . . . 514
10.86.4.53SetVelocity . . . 514
10.86.4.54Update . . . 514
10.86.4.55UpdateParameters . . . 514
10.86.5 Member Data Documentation . . . 515
10.86.5.1 anchorLink . . . 515
10.86.5.2 anchorPos . . . 515
10.86.5.3 anchorPose . . . 515
10.86.5.4 applyDamping . . . 515

10.86.5.5 childLink . . . 515
10.86.5.6 dampingCoefficient . . . 515
10.86.5.7 effortLimit . . . 515
10.86.5.8 inertiaRatio . . . 515
10.86.5.9 lowerLimit . . . 515
10.86.5.10 model . . . 516
10.86.5.11 parentLink . . . 516
10.86.5.12 provideFeedback . . . 516
10.86.5.13 upperLimit . . . 516
10.86.5.14 useCFMDamping . . . 516
10.86.5.15 velocityLimit . . . 516
10.86.5.16 wrench . . . 516
10.87 Joint_TEST Class Reference . . . 516
10.87.1 Constructor & Destructor Documentation . . . 518
10.87.1.1 Joint_TEST . . . 518
10.87.2 Member Function Documentation . . . 518
10.87.2.1 ForceTorque1 . . . 518
10.87.2.2 ForceTorque2 . . . 518
10.87.2.3 GetForceTorqueWithAppliedForce . . . 518
10.87.2.4 JointCreationDestructionTest . . . 519
10.87.2.5 JointTorqueTest . . . 519
10.87.2.6 SetUp . . . 519
10.87.2.7 SpawnJoint . . . 519
10.87.2.8 SpawnJoint . . . 519
10.87.2.9 SpawnJointRotational . . . 520
10.87.2.10 SpawnJointRotationalWorld . . . 520
10.87.2.11 SpawnJointTypes . . . 520
10.87.3 Member Data Documentation . . . 520
10.87.3.1 jointType . . . 520
10.87.3.2 physicsEngine . . . 520
10.88 gazebo::physics::JointController Class Reference . . . 521
10.88.1 Detailed Description . . . 521
10.88.2 Constructor & Destructor Documentation . . . 521
10.88.2.1 JointController . . . 521
10.88.3 Member Function Documentation . . . 521
10.88.3.1 AddJoint . . . 521
10.88.3.2 Reset . . . 522

10.88.3.3 SetJointPosition . . . 522
10.88.3.4 SetJointPosition . . . 522
10.88.3.5 SetJointPositions . . . 522
10.88.3.6 Update . . . 522
10.89 gazebo::physics::JointState Class Reference . . . 522
10.89.1 Detailed Description . . . 524
10.89.2 Constructor & Destructor Documentation . . . 524
10.89.2.1 JointState . . . 524
10.89.2.2 JointState . . . 524
10.89.2.3 JointState . . . 524
10.89.2.4 JointState . . . 524
10.89.2.5 ∼JointState . . . 525
10.89.3 Member Function Documentation . . . 525
10.89.3.1 FillSDF . . . 525
10.89.3.2 GetAngle . . . 525
10.89.3.3 GetAngleCount . . . 525
10.89.3.4 GetAngles . . . 525
10.89.3.5 IsZero . . . 526
10.89.3.6 Load . . . 526
10.89.3.7 Load . . . 526
10.89.3.8 operator+ . . . 526
10.89.3.9 operator- . . . 526
10.89.3.10 operator= . . . 527
10.89.4 Friends And Related Function Documentation . . . 527
10.89.4.1 operator<< . . . 527
10.90 gazebo::rendering::JointVisual Class Reference . . . 527
10.90.1 Detailed Description . . . 528
10.90.2 Constructor & Destructor Documentation . . . 528
10.90.2.1 JointVisual . . . 528
10.90.2.2 ∼JointVisual . . . 529
10.90.3 Member Function Documentation . . . 529
10.90.3.1 Load . . . 529
10.91 gazebo::physics::JointWrench Class Reference . . . 529
10.91.1 Detailed Description . . . 529
10.91.2 Member Function Documentation . . . 530
10.91.2.1 operator+ . . . 530
10.91.2.2 operator- . . . 530

10.91.2.3 operator= . . . 530
10.91.3 Member Data Documentation . . . 530
10.91.3.1 body1Force . . . 530
10.91.3.2 body1Torque . . . 531
10.91.3.3 body2Force . . . 531
10.91.3.4 body2Torque . . . 531
10.92gazebo::common::KeyEvent Class Reference . . . 531
10.92.1 Detailed Description . . . 531
10.92.2 Member Enumeration Documentation . . . 532
10.92.2.1 EventType . . . 532
10.92.3 Constructor & Destructor Documentation . . . 532
10.92.3.1 KeyEvent . . . 532
10.92.4 Member Data Documentation . . . 532
10.92.4.1 key . . . 532
10.92.4.2 type . . . 532
10.93gazebo::common::KeyFrame Class Reference . . . 532
10.93.1 Detailed Description . . . 533
10.93.2 Constructor & Destructor Documentation . . . 533
10.93.2.1 KeyFrame . . . 533
10.93.2.2 ∼KeyFrame . . . 533
10.93.3 Member Function Documentation . . . 533
10.93.3.1 GetTime . . . 533
10.93.4 Member Data Documentation . . . 533
10.93.4.1 time . . . 533
10.94gazebo::rendering::LaserVisual Class Reference . . . 534
10.94.1 Detailed Description . . . 534
10.94.2 Constructor & Destructor Documentation . . . 535
10.94.2.1 LaserVisual . . . 535
10.94.2.2 ∼LaserVisual . . . 535
10.94.3 Member Function Documentation . . . 535
10.94.3.1 SetEmissive . . . 535
10.95gazebo::rendering::Light Class Reference . . . 535
10.95.1 Detailed Description . . . 537
10.95.2 Constructor & Destructor Documentation . . . 537
10.95.2.1 Light . . . 537
10.95.2.2 ∼Light . . . 537
10.95.3 Member Function Documentation . . . 537

10.95.3.1 FillMsg . . . 537
10.95.3.2 GetDiffuseColor . . . 537
10.95.3.3 GetDirection . . . 537
10.95.3.4 GetName . . . 538
10.95.3.5 GetPosition . . . 538
10.95.3.6 GetSpecularColor . . . 538
10.95.3.7 GetType . . . 538
10.95.3.8 GetVisible . . . 538
10.95.3.9 Load . . . 538
10.95.3.10 Load . . . 539
10.95.3.11 LoadFromMsg . . . 539
10.95.3.12 OnPoseChange . . . 539
10.95.3.13 SetAttenuation . . . 539
10.95.3.14 SetCastShadows . . . 539
10.95.3.15 SetDiffuseColor . . . 539
10.95.3.16 SetDirection . . . 540
10.95.3.17 SetLightType . . . 540
10.95.3.18 SetName . . . 540
10.95.3.19 SetPosition . . . 540
10.95.3.20 SetRange . . . 540
10.95.3.21 SetSelected . . . 540
10.95.3.22 SetSpecularColor . . . 541
10.95.3.23 SetSpotFalloff . . . 541
10.95.3.24 SetSpotInnerAngle . . . 541
10.95.3.25 SetSpotOuterAngle . . . 541
10.95.3.26 ShowVisual . . . 541
10.95.3.27 ToggleShowVisual . . . 541
10.95.3.28 UpdateFromMsg . . . 541
10.96 gazebo::physics::Link Class Reference . . . 542
10.96.1 Detailed Description . . . 547
10.96.2 Member Typedef Documentation . . . 547
10.96.2.1 Visuals_M . . . 547
10.96.3 Constructor & Destructor Documentation . . . 547
10.96.3.1 Link . . . 547
10.96.3.2 ∼Link . . . 547
10.96.4 Member Function Documentation . . . 547
10.96.4.1 AddChildJoint . . . 547

10.96.4.2 AddForce . . . 547
10.96.4.3 AddForceAtRelativePosition . . . 547
10.96.4.4 AddForceAtWorldPosition . . . 548
10.96.4.5 AddParentJoint . . . 548
10.96.4.6 AddRelativeForce . . . 548
10.96.4.7 AddRelativeTorque . . . 548
10.96.4.8 AddTorque . . . 548
10.96.4.9 AttachStaticModel . . . 549
10.96.4.10 ConnectEnabled . . . 549
10.96.4.11 DetachAllStaticModels . . . 549
10.96.4.12 DetachStaticModel . . . 549
10.96.4.13 DisconnectEnabled . . . 549
10.96.4.14 FillMsg . . . 550
10.96.4.15 Fini . . . 550
10.96.4.16 GetAngularDamping . . . 550
10.96.4.17 GetBoundingBox . . . 550
10.96.4.18 GetChildJoints . . . 550
10.96.4.19 GetChildJointsLinks . . . 550
10.96.4.20 GetCollision . . . 551
10.96.4.21 GetCollision . . . 551
10.96.4.22 GetCollisions . . . 551
10.96.4.23 GetEnabled . . . 551
10.96.4.24 GetGravityMode . . . 551
10.96.4.25 GetInertial . . . 552
10.96.4.26 GetKinematic . . . 552
10.96.4.27 GetLinearDamping . . . 552
10.96.4.28 GetModel . . . 552
10.96.4.29 GetParentJoints . . . 552
10.96.4.30 GetParentJointsLinks . . . 552
10.96.4.31 GetRelativeAngularAccel . . . 553
10.96.4.32 GetRelativeAngularVel . . . 553
10.96.4.33 GetRelativeForce . . . 553
10.96.4.34 GetRelativeLinearAccel . . . 553
10.96.4.35 GetRelativeLinearVel . . . 553
10.96.4.36 GetRelativeTorque . . . 554
10.96.4.37 GetSelfCollide . . . 554
10.96.4.38 GetSensorCount . . . 554

10.96.4.39GetSensorName . . . 554
10.96.4.40GetWorldAngularAccel . . . 554
10.96.4.41GetWorldCoGLinearVel . . . 555
10.96.4.42GetWorldCoGPose . . . 555
10.96.4.43GetWorldForce . . . 555
10.96.4.44GetWorldLinearAccel . . . 555
10.96.4.45GetWorldLinearVel . . . 555
10.96.4.46GetWorldLinearVel . . . 556
10.96.4.47GetWorldTorque . . . 556
10.96.4.48Init . . . 556
10.96.4.49Load . . . 556
10.96.4.50OnPoseChange . . . 557
10.96.4.51ProcessMsg . . . 557
10.96.4.52RemoveChild . . . 557
10.96.4.53RemoveChildJoint . . . 557
10.96.4.54RemoveCollision . . . 557
10.96.4.55RemoveParentJoint . . . 557
10.96.4.56Reset . . . 557
10.96.4.57ResetPhysicsStates . . . 558
10.96.4.58SetAngularAccel . . . 558
10.96.4.59SetAngularDamping . . . 558
10.96.4.60SetAngularVel . . . 558
10.96.4.61SetAutoDisable . . . 558
10.96.4.62SetCollideMode . . . 558
10.96.4.63SetEnabled . . . 559
10.96.4.64SetForce . . . 559
10.96.4.65SetGravityMode . . . 559
10.96.4.66SetInertial . . . 559
10.96.4.67SetKinematic . . . 559
10.96.4.68SetLaserRetro . . . 560
10.96.4.69SetLinearAccel . . . 560
10.96.4.70SetLinearDamping . . . 560
10.96.4.71SetLinearVel . . . 560
10.96.4.72SetLinkStatic . . . 560
10.96.4.73SetPublishData . . . 561
10.96.4.74SetScale . . . 561
10.96.4.75SetSelected . . . 561

10.96.4.76 SetSelfCollide . . . 561
10.96.4.77 SetState . . . 561
10.96.4.78 SetTorque . . . 561
10.96.4.79 Update . . . 562
10.96.4.80 UpdateMass . . . 562
10.96.4.81 UpdateParameters . . . 562
10.96.4.82 UpdateSurface . . . 562
10.96.5 Member Data Documentation . . . 562
10.96.5.1 angularAccel . . . 562
10.96.5.2 attachedModelsOffset . . . 562
10.96.5.3 cgVisuals . . . 562
10.96.5.4 inertial . . . 563
10.96.5.5 linearAccel . . . 563
10.96.5.6 visuals . . . 563
10.97 gazebo::physics::LinkState Class Reference . . . 563
10.97.1 Detailed Description . . . 565
10.97.2 Constructor & Destructor Documentation . . . 565
10.97.2.1 LinkState . . . 565
10.97.2.2 LinkState . . . 565
10.97.2.3 LinkState . . . 565
10.97.2.4 LinkState . . . 565
10.97.2.5 ~LinkState . . . 566
10.97.3 Member Function Documentation . . . 566
10.97.3.1 FillSDF . . . 566
10.97.3.2 GetAcceleration . . . 566
10.97.3.3 GetCollisionState . . . 566
10.97.3.4 GetCollisionState . . . 566
10.97.3.5 GetCollisionStateCount . . . 567
10.97.3.6 GetCollisionStates . . . 567
10.97.3.7 GetPose . . . 567
10.97.3.8 GetVelocity . . . 567
10.97.3.9 GetWrench . . . 568
10.97.3.10 IsZero . . . 568
10.97.3.11 Load . . . 568
10.97.3.12 Load . . . 568
10.97.3.13 operator+ . . . 568
10.97.3.14 operator- . . . 569

10.97.3.15 operator= . . . 569
10.97.3.16 SetRealTime . . . 569
10.97.3.17 SetSimTime . . . 569
10.97.3.18 SetWallTime . . . 570
10.97.4 Friends And Related Function Documentation . . . 570
10.97.4.1 operator<< . . . 570
10.98 gazebo::util::LogPlay Class Reference . . . 570
10.98.1 Member Function Documentation . . . 571
10.98.1.1 GetChunk . . . 571
10.98.1.2 GetChunkCount . . . 571
10.98.1.3 GetEncoding . . . 572
10.98.1.4 GetGazeboVersion . . . 572
10.98.1.5 GetHeader . . . 572
10.98.1.6 GetLogVersion . . . 572
10.98.1.7 GetRandSeed . . . 572
10.98.1.8 IsOpen . . . 572
10.98.1.9 Open . . . 573
10.98.1.10 Step . . . 573
10.99 Logplay Class Reference . . . 573
10.99.1 Detailed Description . . . 573
10.100 gazebo::util::LogRecord Class Reference . . . 573
10.100.1 Detailed Description . . . 575
10.100.2 Member Function Documentation . . . 575
10.100.2.1 Add . . . 575
10.100.2.2 Fini . . . 576
10.100.2.3 GetBasePath . . . 576
10.100.2.4 GetBufferSize . . . 576
10.100.2.5 GetEncoding . . . 576
10.100.2.6 GetFilename . . . 576
10.100.2.7 GetFileSize . . . 576
10.100.2.8 GetFirstUpdate . . . 577
10.100.2.9 GetPaused . . . 577
10.100.2.10 GetRunning . . . 577
10.100.2.11 GetRunTime . . . 577
10.100.2.12 Init . . . 577
10.100.2.13 IsReadyToStart . . . 578
10.100.2.14 Notify . . . 578

10.100.2.15 Remove . . . 578
10.100.2.16 SetBasePath . . . 578
10.100.2.17 SetPaused . . . 578
10.100.2.18 Start . . . 579
10.100.2.19 Stop . . . 579
10.100.2.20 Write . . . 579
10.101 gazebo::physics::MapShape Class Reference . . . 579
10.101.1 Detailed Description . . . 581
10.101.2 Constructor & Destructor Documentation . . . 581
10.101.2.1 MapShape . . . 581
10.101.2.2 ~MapShape . . . 581
10.101.3 Member Function Documentation . . . 581
10.101.3.1 FillMsg . . . 581
10.101.3.2 GetGranularity . . . 581
10.101.3.3 GetHeight . . . 582
10.101.3.4 GetScale . . . 582
10.101.3.5 GetThreshold . . . 582
10.101.3.6 GetURI . . . 582
10.101.3.7 Init . . . 582
10.101.3.8 Load . . . 582
10.101.3.9 ProcessMsg . . . 583
10.101.3.10 SetScale . . . 583
10.101.3.11 Update . . . 583
10.102 gazebo::Master Class Reference . . . 583
10.102.1 Detailed Description . . . 584
10.102.2 Constructor & Destructor Documentation . . . 584
10.102.2.1 Master . . . 584
10.102.2.2 ~Master . . . 584
10.102.3 Member Function Documentation . . . 584
10.102.3.1 Fini . . . 584
10.102.3.2 Init . . . 584
10.102.3.3 Run . . . 584
10.102.3.4 RunOnce . . . 584
10.102.3.5 RunThread . . . 585
10.102.3.6 Stop . . . 585
10.103 gazebo::common::Material Class Reference . . . 585
10.103.1 Detailed Description . . . 587

10.103.2 Member Enumeration Documentation . . . 587
10.103.2.1 BlendMode . . . 587
10.103.2.2 ShadeMode . . . 588
10.103.3 Constructor & Destructor Documentation . . . 588
10.103.3.1 Material . . . 588
10.103.3.2 ~Material . . . 588
10.103.3.3 Material . . . 588
10.103.4 Member Function Documentation . . . 588
10.103.4.1 GetAmbient . . . 588
10.103.4.2 GetBlendFactors . . . 588
10.103.4.3 GetBlendMode . . . 589
10.103.4.4 GetDepthWrite . . . 589
10.103.4.5 GetDiffuse . . . 589
10.103.4.6 GetEmissive . . . 589
10.103.4.7 GetLighting . . . 589
10.103.4.8 GetName . . . 589
10.103.4.9 GetPointSize . . . 590
10.103.4.10 GetShadeMode . . . 590
10.103.4.11 GetShininess . . . 590
10.103.4.12 GetSpecular . . . 590
10.103.4.13 GetTextureImage . . . 590
10.103.4.14 GetTransparency . . . 590
10.103.4.15 SetAmbient . . . 591
10.103.4.16 SetBlendFactors . . . 591
10.103.4.17 SetBlendMode . . . 591
10.103.4.18 SetDepthWrite . . . 591
10.103.4.19 SetDiffuse . . . 591
10.103.4.20 SetEmissive . . . 591
10.103.4.21 SetLighting . . . 592
10.103.4.22 SetPointSize . . . 592
10.103.4.23 SetShadeMode . . . 592
10.103.4.24 SetShininess . . . 592
10.103.4.25 SetSpecular . . . 592
10.103.4.26 SetTextureImage . . . 592
10.103.4.27 SetTextureImage . . . 593
10.103.4.28 SetTransparency . . . 593
10.103.5 Friends And Related Function Documentation . . . 593

10.103.5.1 operator<< . . . 593
10.103.6 Member Data Documentation . . . 593
10.103.6.1 ambient . . . 593
10.103.6.2 blendMode . . . 593
10.103.6.3 BlendModeStr . . . 593
10.103.6.4 diffuse . . . 593
10.103.6.5 emissive . . . 593
10.103.6.6 name . . . 593
10.103.6.7 pointSize . . . 594
10.103.6.8 shadeMode . . . 594
10.103.6.9 ShadeModeStr . . . 594
10.103.6.10 shininess . . . 594
10.103.6.11 specular . . . 594
10.103.6.12 texImage . . . 594
10.103.6.13 transparency . . . 594
10.104 gazebo::math::Matrix3 Class Reference . . . 594
10.104.1 Detailed Description . . . 595
10.104.2 Constructor & Destructor Documentation . . . 595
10.104.2.1 Matrix3 . . . 595
10.104.2.2 Matrix3 . . . 596
10.104.2.3 Matrix3 . . . 596
10.104.2.4 ~Matrix3 . . . 596
10.104.3 Member Function Documentation . . . 596
10.104.3.1 operator* . . . 596
10.104.3.2 operator* . . . 596
10.104.3.3 operator+ . . . 597
10.104.3.4 operator- . . . 597
10.104.3.5 operator== . . . 597
10.104.3.6 operator[] . . . 597
10.104.3.7 operator[] . . . 597
10.104.3.8 SetCol . . . 598
10.104.3.9 SetFromAxes . . . 598
10.104.3.10 SetFromAxis . . . 598
10.104.4 Friends And Related Function Documentation . . . 598
10.104.4.1 operator* . . . 598
10.104.4.2 operator<< . . . 598
10.104.5 Member Data Documentation . . . 599

10.104.5.1 m . . . 599
10.105 gazebo::math::Matrix4 Class Reference . . . 599
10.105.1 Detailed Description . . . 600
10.105.2 Constructor & Destructor Documentation . . . 601
10.105.2.1 Matrix4 . . . 601
10.105.2.2 Matrix4 . . . 601
10.105.2.3 Matrix4 . . . 601
10.105.2.4 ~Matrix4 . . . 601
10.105.3 Member Function Documentation . . . 601
10.105.3.1 GetAsPose . . . 601
10.105.3.2 GetEulerRotation . . . 602
10.105.3.3 GetRotation . . . 602
10.105.3.4 GetTranslation . . . 602
10.105.3.5 Inverse . . . 602
10.105.3.6 IsAffine . . . 602
10.105.3.7 operator* . . . 602
10.105.3.8 operator* . . . 603
10.105.3.9 operator* . . . 603
10.105.3.10 operator= . . . 603
10.105.3.11 operator= . . . 603
10.105.3.12 operator== . . . 604
10.105.3.13 operator[] . . . 604
10.105.3.14 operator[] . . . 604
10.105.3.15 Set . . . 604
10.105.3.16 SetScale . . . 605
10.105.3.17 SetTranslate . . . 605
10.105.3.18 TransformAffine . . . 605
10.105.4 Friends And Related Function Documentation . . . 605
10.105.4.1 operator<< . . . 606
10.105.5 Member Data Documentation . . . 606
10.105.5.1 IDENTITY . . . 606
10.105.5.2 m . . . 606
10.105.5.3 ZERO . . . 606
10.106 gazebo::common::Mesh Class Reference . . . 606
10.106.1 Detailed Description . . . 608
10.106.2 Constructor & Destructor Documentation . . . 608
10.106.2.1 Mesh . . . 608

10.106.2.2 ~Mesh . . . 608
10.106.3 Member Function Documentation . . . 608
10.106.3.1 AddMaterial . . . 608
10.106.3.2 AddSubMesh . . . 608
10.106.3.3 Center . . . 608
10.106.3.4 FillArrays . . . 609
10.106.3.5 GenSphericalTexCoord . . . 609
10.106.3.6 GetAABB . . . 609
10.106.3.7 GetIndexCount . . . 609
10.106.3.8 GetMaterial . . . 609
10.106.3.9 GetMaterialCount . . . 610
10.106.3.10 GetMax . . . 610
10.106.3.11 GetMin . . . 610
10.106.3.12 GetName . . . 610
10.106.3.13 GetNormalCount . . . 610
10.106.3.14 GetPath . . . 610
10.106.3.15 GetSkeleton . . . 611
10.106.3.16 GetSubMesh . . . 611
10.106.3.17 GetSubMesh . . . 611
10.106.3.18 GetSubMeshCount . . . 611
10.106.3.19 GetTexCoordCount . . . 611
10.106.3.20 GetVertexCount . . . 612
10.106.3.21 HasSkeleton . . . 612
10.106.3.22 RecalculateNormals . . . 612
10.106.3.23 Scale . . . 612
10.106.3.24 SetName . . . 612
10.106.3.25 SetPath . . . 612
10.106.3.26 SetScale . . . 612
10.106.3.27 SetSkeleton . . . 613
10.106.3.28 Translate . . . 613
10.107 gazebo::common::MeshCSG Class Reference . . . 613
10.107.1 Detailed Description . . . 613
10.107.2 Member Enumeration Documentation . . . 613
10.107.2.1 BooleanOperation . . . 613
10.107.3 Constructor & Destructor Documentation . . . 614
10.107.3.1 MeshCSG . . . 614
10.107.3.2 ~MeshCSG . . . 614

10.107.4 Member Function Documentation . . . 614
10.107.4.1 CreateBoolean . . . 614
10.108 gazebo::common::MeshLoader Class Reference . . . 614
10.108.1 Detailed Description . . . 615
10.108.2 Constructor & Destructor Documentation . . . 615
10.108.2.1 MeshLoader . . . 615
10.108.2.2 ~MeshLoader . . . 615
10.108.3 Member Function Documentation . . . 615
10.108.3.1 Load . . . 615
10.109 gazebo::common::MeshManager Class Reference . . . 616
10.109.1 Detailed Description . . . 617
10.109.2 Member Function Documentation . . . 617
10.109.2.1 AddMesh . . . 617
10.109.2.2 CreateBox . . . 617
10.109.2.3 CreateCamera . . . 618
10.109.2.4 CreateCone . . . 618
10.109.2.5 CreateCylinder . . . 618
10.109.2.6 CreatePlane . . . 618
10.109.2.7 CreatePlane . . . 619
10.109.2.8 CreateSphere . . . 619
10.109.2.9 CreateTube . . . 619
10.109.2.10 GenSphericalTexCoord . . . 619
10.109.2.11 GetMesh . . . 619
10.109.2.12 GetMeshAABB . . . 620
10.109.2.13 HasMesh . . . 620
10.109.2.14 IsValidFilename . . . 620
10.109.2.15 Load . . . 620
10.110 gazebo::physics::MeshShape Class Reference . . . 621
10.110.1 Detailed Description . . . 622
10.110.2 Constructor & Destructor Documentation . . . 622
10.110.2.1 MeshShape . . . 622
10.110.2.2 ~MeshShape . . . 622
10.110.3 Member Function Documentation . . . 622
10.110.3.1 FillMsg . . . 622
10.110.3.2 GetMeshURI . . . 623
10.110.3.3 GetSize . . . 623
10.110.3.4 Init . . . 623

10.110.3.5 ProcessMsg . . . 623
10.110.3.6 SetMesh . . . 623
10.110.3.7 SetScale . . . 623
10.110.3.8 Update . . . 624
10.110.4 Member Data Documentation . . . 624
10.110.4.1 mesh . . . 624
10.110.4.2 submesh . . . 624
10.111 gazebo::physics::Model Class Reference . . . 624
10.111.1 Detailed Description . . . 628
10.111.2 Constructor & Destructor Documentation . . . 628
10.111.2.1 Model . . . 628
10.111.2.2 ~Model . . . 628
10.111.3 Member Function Documentation . . . 628
10.111.3.1 AttachStaticModel . . . 628
10.111.3.2 DetachStaticModel . . . 629
10.111.3.3 FillMsg . . . 629
10.111.3.4 Fini . . . 629
10.111.3.5 GetAutoDisable . . . 629
10.111.3.6 GetBoundingBox . . . 629
10.111.3.7 GetGripper . . . 629
10.111.3.8 GetGripperCount . . . 630
10.111.3.9 GetJoint . . . 630
10.111.3.10 GetJointController . . . 630
10.111.3.11 GetJointCount . . . 630
10.111.3.12 GetJoints . . . 630
10.111.3.13 GetLink . . . 631
10.111.3.14 GetLinks . . . 631
10.111.3.15 GetPluginCount . . . 631
10.111.3.16 GetRelativeAngularAccel . . . 631
10.111.3.17 GetRelativeAngularVel . . . 631
10.111.3.18 GetRelativeLinearAccel . . . 632
10.111.3.19 GetRelativeLinearVel . . . 632
10.111.3.20 GetSDF . . . 632
10.111.3.21 GetSensorCount . . . 632
10.111.3.22 GetWorldAngularAccel . . . 632
10.111.3.23 GetWorldAngularVel . . . 633
10.111.3.24 GetWorldLinearAccel . . . 633

10.111.3.25 SetWorldLinearVel . . . 633
10.111.3.26 Init . . . 633
10.111.3.27 Load . . . 633
10.111.3.28 LoadJoints . . . 634
10.111.3.29 LoadPlugins . . . 634
10.111.3.30 OnPoseChange . . . 634
10.111.3.31 ProcessMsg . . . 634
10.111.3.32 RemoveChild . . . 634
10.111.3.33 Reset . . . 634
10.111.3.34 SetAngularAccel . . . 634
10.111.3.35 SetAngularVel . . . 634
10.111.3.36 SetAutoDisable . . . 635
10.111.3.37 SetCollideMode . . . 635
10.111.3.38 SetEnabled . . . 635
10.111.3.39 SetGravityMode . . . 635
10.111.3.40 SetJointAnimation . . . 635
10.111.3.41 SetJointPosition . . . 636
10.111.3.42 SetJointPositions . . . 636
10.111.3.43 SetLaserRetro . . . 636
10.111.3.44 SetLinearAccel . . . 636
10.111.3.45 SetLinearVel . . . 636
10.111.3.46 SetLinkWorldPose . . . 637
10.111.3.47 SetLinkWorldPose . . . 637
10.111.3.48 SetScale . . . 637
10.111.3.49 SetState . . . 637
10.111.3.50 StopAnimation . . . 637
10.111.3.51 Update . . . 637
10.111.3.52 UpdateParameters . . . 638
10.111.4 Member Data Documentation . . . 638
10.111.4.1 attachedModels . . . 638
10.111.4.2 attachedModelsOffset . . . 638
10.111.4.3 jointPub . . . 638
10.112 gazebo::common::ModelDatabase Class Reference . . . 638
10.112.1 Detailed Description . . . 639
10.113 gazebo::ModelPlugin Class Reference . . . 639
10.113.1 Detailed Description . . . 640
10.113.2 Constructor & Destructor Documentation . . . 640

10.113.2.1 ModelPlugin . . . 640
10.113.2.2 ~ModelPlugin . . . 640
10.113.3 Member Function Documentation . . . 641
10.113.3.1 Init . . . 641
10.113.3.2 Load . . . 641
10.113.3.3 Reset . . . 641
10.114 gazebo::physics::ModelState Class Reference . . . 641
10.114.1 Detailed Description . . . 643
10.114.2 Constructor & Destructor Documentation . . . 643
10.114.2.1 ModelState . . . 643
10.114.2.2 ModelState . . . 643
10.114.2.3 ModelState . . . 643
10.114.2.4 ModelState . . . 644
10.114.2.5 ~ModelState . . . 644
10.114.3 Member Function Documentation . . . 644
10.114.3.1 FillSDF . . . 644
10.114.3.2 GetJointState . . . 644
10.114.3.3 GetJointState . . . 644
10.114.3.4 GetJointStateCount . . . 645
10.114.3.5 GetJointStates . . . 645
10.114.3.6 GetJointStates . . . 645
10.114.3.7 GetLinkState . . . 645
10.114.3.8 GetLinkStateCount . . . 646
10.114.3.9 GetLinkStates . . . 646
10.114.3.10 GetLinkStates . . . 646
10.114.3.11 GetPose . . . 646
10.114.3.12 HasJointState . . . 647
10.114.3.13 HasLinkState . . . 647
10.114.3.14 IsZero . . . 647
10.114.3.15 Load . . . 647
10.114.3.16 Load . . . 647
10.114.3.17 operator+ . . . 648
10.114.3.18 operator- . . . 648
10.114.3.19 operator= . . . 648
10.114.3.20 SetRealTime . . . 648
10.114.3.21 SetSimTime . . . 649
10.114.3.22 SetWallTime . . . 649

10.114.4 Friends And Related Function Documentation . . . 649
10.114.4.1 operator<< . . . 649
10.115 gazebo::common::MouseEvent Class Reference . . . 649
10.115.1 Detailed Description . . . 650
10.115.2 Member Enumeration Documentation . . . 651
10.115.2.1 Buttons . . . 651
10.115.2.2 EventType . . . 651
10.115.3 Constructor & Destructor Documentation . . . 651
10.115.3.1 MouseEvent . . . 651
10.115.4 Member Data Documentation . . . 651
10.115.4.1 alt . . . 651
10.115.4.2 button . . . 651
10.115.4.3 buttons . . . 651
10.115.4.4 control . . . 651
10.115.4.5 dragging . . . 652
10.115.4.6 moveScale . . . 652
10.115.4.7 pos . . . 652
10.115.4.8 pressPos . . . 652
10.115.4.9 prevPos . . . 652
10.115.4.10 scroll . . . 652
10.115.4.11 shift . . . 652
10.115.4.12 type . . . 652
10.116 gazebo::rendering::MovableText Class Reference . . . 652
10.116.1 Detailed Description . . . 654
10.116.2 Member Enumeration Documentation . . . 654
10.116.2.1 HorizAlign . . . 654
10.116.2.2 VertAlign . . . 655
10.116.3 Constructor & Destructor Documentation . . . 655
10.116.3.1 MovableText . . . 655
10.116.3.2 ~MovableText . . . 655
10.116.4 Member Function Documentation . . . 655
10.116.4.1 _setupGeometry . . . 655
10.116.4.2 _updateColors . . . 655
10.116.4.3 GetAABB . . . 655
10.116.4.4 GetBaseline . . . 655
10.116.4.5 getBoundingRadius . . . 655
10.116.4.6 GetCharHeight . . . 655

10.116.4.7 GetColor . . . 656
10.116.4.8 GetFont . . . 656
10.116.4.9 getLights . . . 656
10.116.4.10 GetMaterial . . . 656
10.116.4.11 GetRenderOperation . . . 656
10.116.4.12 GetShowOnTop . . . 656
10.116.4.13 GetSpaceWidth . . . 656
10.116.4.14 GetSquaredViewDepth . . . 656
10.116.4.15 GetText . . . 656
10.116.4.16 getWorldTransforms . . . 657
10.116.4.17 Load . . . 657
10.116.4.18 SetBaseline . . . 657
10.116.4.19 SetCharHeight . . . 657
10.116.4.20 SetColor . . . 657
10.116.4.21 SetFontName . . . 657
10.116.4.22 SetShowOnTop . . . 658
10.116.4.23 SetSpaceWidth . . . 658
10.116.4.24 SetText . . . 658
10.116.4.25 SetTextAlignment . . . 658
10.116.4.26 Update . . . 658
10.116.4.27 VisitRenderables . . . 658
10.117 gazebo::msgs::MsgFactory Class Reference . . . 658
10.117.1 Detailed Description . . . 659
10.117.2 Member Function Documentation . . . 659
10.117.2.1 GetMsgTypes . . . 659
10.117.2.2 NewMsg . . . 659
10.117.2.3 RegisterMsg . . . 659
10.118 gazebo::sensors::MultiCameraSensor Class Reference . . . 660
10.118.1 Detailed Description . . . 661
10.118.2 Constructor & Destructor Documentation . . . 661
10.118.2.1 MultiCameraSensor . . . 661
10.118.2.2 ~MultiCameraSensor . . . 661
10.118.3 Member Function Documentation . . . 661
10.118.3.1 Fini . . . 661
10.118.3.2 GetCamera . . . 662
10.118.3.3 GetCameraCount . . . 662
10.118.3.4 GetImageData . . . 662

10.118.3.5 GetImageHeight . . . 662
10.118.3.6 GetImageWidth . . . 663
10.118.3.7 GetTopic . . . 663
10.118.3.8 Init . . . 663
10.118.3.9 IsActive . . . 663
10.118.3.10 Load . . . 664
10.118.3.11 SaveFrame . . . 664
10.118.3.12 UpdateImpl . . . 664
10.119 gazebo::physics::MultiRayShape Class Reference . . . 664
10.119.1 Detailed Description . . . 667
10.119.2 Constructor & Destructor Documentation . . . 667
10.119.2.1 MultiRayShape . . . 667
10.119.2.2 ~MultiRayShape . . . 667
10.119.3 Member Function Documentation . . . 667
10.119.3.1 AddRay . . . 667
10.119.3.2 ConnectNewLaserScans . . . 668
10.119.3.3 DisconnectNewLaserScans . . . 668
10.119.3.4 FillMsg . . . 668
10.119.3.5 GetFiducial . . . 668
10.119.3.6 GetMaxAngle . . . 668
10.119.3.7 GetMaxRange . . . 669
10.119.3.8 GetMinAngle . . . 669
10.119.3.9 GetMinRange . . . 669
10.119.3.10 GetRange . . . 669
10.119.3.11 GetResRange . . . 669
10.119.3.12 GetRetro . . . 670
10.119.3.13 GetSampleCount . . . 670
10.119.3.14 GetScanResolution . . . 670
10.119.3.15 GetVerticalMaxAngle . . . 670
10.119.3.16 GetVerticalMinAngle . . . 670
10.119.3.17 GetVerticalSampleCount . . . 670
10.119.3.18 GetVerticalScanResolution . . . 671
10.119.3.19 Init . . . 671
10.119.3.20 ProcessMsg . . . 671
10.119.3.21 SetScale . . . 671
10.119.3.22 Update . . . 671
10.119.3.23 UpdateRays . . . 671

10.119.4 Member Data Documentation . . . 671
10.119.4.1 horzElem . . . 672
10.119.4.2 newLaserScans . . . 672
10.119.4.3 offset . . . 672
10.119.4.4 rangeElem . . . 672
10.119.4.5 rayElem . . . 672
10.119.4.6 rays . . . 672
10.119.4.7 scanElem . . . 672
10.119.4.8 vertElem . . . 672
10.120 gazebo::transport::Node Class Reference . . . 672
10.120.1 Detailed Description . . . 674
10.120.2 Constructor & Destructor Documentation . . . 674
10.120.2.1 Node . . . 674
10.120.2.2 ~Node . . . 674
10.120.3 Member Function Documentation . . . 674
10.120.3.1 Advertise . . . 674
10.120.3.2 DecodeTopicName . . . 675
10.120.3.3 EncodeTopicName . . . 675
10.120.3.4 Fini . . . 675
10.120.3.5 GetId . . . 675
10.120.3.6 GetMsgType . . . 676
10.120.3.7 GetTopicNamespace . . . 676
10.120.3.8 HandleData . . . 676
10.120.3.9 HandleMessage . . . 676
10.120.3.10 HasLatchedSubscriber . . . 676
10.120.3.11 Init . . . 677
10.120.3.12 InsertLatchedMsg . . . 677
10.120.3.13 InsertLatchedMsg . . . 677
10.120.3.14 ProcessIncoming . . . 677
10.120.3.15 ProcessPublishers . . . 677
10.120.3.16 Publish . . . 678
10.120.3.17 RemoveCallback . . . 678
10.120.3.18 Subscribe . . . 678
10.120.3.19 Subscribe . . . 678
10.120.3.20 Subscribe . . . 679
10.120.3.21 Subscribe . . . 679
10.121 gazebo::common::NodeAnimation Class Reference . . . 679

10.121.1 Detailed Description . . . 680
10.121.2 Constructor & Destructor Documentation . . . 680
10.121.2.1 NodeAnimation . . . 680
10.121.2.2 ~NodeAnimation . . . 681
10.121.3 Member Function Documentation . . . 681
10.121.3.1 AddKeyFrame . . . 681
10.121.3.2 AddKeyFrame . . . 681
10.121.3.3 GetFrameAt . . . 681
10.121.3.4 GetFrameCount . . . 681
10.121.3.5 GetKeyFrame . . . 681
10.121.3.6 GetKeyFrame . . . 682
10.121.3.7 GetLength . . . 682
10.121.3.8 GetName . . . 682
10.121.3.9 GetTimeAtX . . . 682
10.121.3.10 Scale . . . 682
10.121.3.11 SetName . . . 683
10.121.4 Member Data Documentation . . . 683
10.121.4.1 keyFrames . . . 683
10.121.4.2 length . . . 683
10.121.4.3 name . . . 683
10.122 gazebo::common::NodeAssignment Struct Reference . . . 683
10.122.1 Detailed Description . . . 684
10.122.2 Member Data Documentation . . . 684
10.122.2.1 nodeIndex . . . 684
10.122.2.2 vertexIndex . . . 684
10.122.2.3 weight . . . 684
10.123 gazebo::common::NodeTransform Class Reference . . . 684
10.123.1 Detailed Description . . . 685
10.123.2 Member Enumeration Documentation . . . 686
10.123.2.1 TransformType . . . 686
10.123.3 Constructor & Destructor Documentation . . . 686
10.123.3.1 NodeTransform . . . 686
10.123.3.2 NodeTransform . . . 686
10.123.3.3 ~NodeTransform . . . 686
10.123.4 Member Function Documentation . . . 686
10.123.4.1 Get . . . 686
10.123.4.2 GetSID . . . 687

10.123.4.3 GetType . . . 687
10.123.4.4 operator() . . . 687
10.123.4.5 operator* . . . 687
10.123.4.6 operator* . . . 687
10.123.4.7 PrintSource . . . 688
10.123.4.8 RecalculateMatrix . . . 688
10.123.4.9 Set . . . 688
10.123.4.10 SetComponent . . . 688
10.123.4.11 SetSID . . . 688
10.123.4.12 SetSourceValues . . . 688
10.123.4.13 SetSourceValues . . . 688
10.123.4.14 SetSourceValues . . . 688
10.123.4.15 SetType . . . 689
10.123.5 Member Data Documentation . . . 689
10.123.5.1 sid . . . 689
10.123.5.2 source . . . 689
10.123.5.3 transform . . . 689
10.123.5.4 type . . . 689
10.124 gazebo::sensors::Noise Class Reference . . . 689
10.124.1 Detailed Description . . . 690
10.124.2 Member Enumeration Documentation . . . 690
10.124.2.1 NoiseType . . . 690
10.124.3 Constructor & Destructor Documentation . . . 690
10.124.3.1 Noise . . . 690
10.124.3.2 ~Noise . . . 690
10.124.4 Member Function Documentation . . . 690
10.124.4.1 Apply . . . 690
10.124.4.2 GetBias . . . 691
10.124.4.3 GetMean . . . 691
10.124.4.4 GetNoiseType . . . 691
10.124.4.5 GetStdDev . . . 691
10.124.4.6 Load . . . 691
10.125 gazebo::common::NumericAnimation Class Reference . . . 692
10.125.1 Detailed Description . . . 692
10.125.2 Constructor & Destructor Documentation . . . 692
10.125.2.1 NumericAnimation . . . 692
10.125.2.2 ~NumericAnimation . . . 693

10.125.3 Member Function Documentation . . . 693
10.125.3.1 CreateKeyFrame . . . 693
10.125.3.2 GetInterpolatedKeyFrame . . . 693
10.126 gazebo::common::NumericKeyFrame Class Reference . . . 693
10.126.1 Detailed Description . . . 694
10.126.2 Constructor & Destructor Documentation . . . 694
10.126.2.1 NumericKeyFrame . . . 694
10.126.2.2 ~NumericKeyFrame . . . 695
10.126.3 Member Function Documentation . . . 695
10.126.3.1 GetValue . . . 695
10.126.3.2 SetValue . . . 695
10.126.4 Member Data Documentation . . . 695
10.126.4.1 value . . . 695
10.127 gazebo::util::OpenAL Class Reference . . . 695
10.127.1 Detailed Description . . . 696
10.127.2 Member Function Documentation . . . 696
10.127.2.1 CreateSink . . . 696
10.127.2.2 CreateSource . . . 697
10.127.2.3 Fini . . . 697
10.127.2.4 Load . . . 697
10.128 gazebo::util::OpenALSink Class Reference . . . 697
10.128.1 Detailed Description . . . 697
10.128.2 Constructor & Destructor Documentation . . . 698
10.128.2.1 OpenALSink . . . 698
10.128.2.2 ~OpenALSink . . . 698
10.128.3 Member Function Documentation . . . 698
10.128.3.1 SetPose . . . 698
10.128.3.2 SetVelocity . . . 698
10.129 gazebo::util::OpenALSource Class Reference . . . 698
10.129.1 Detailed Description . . . 699
10.129.2 Constructor & Destructor Documentation . . . 699
10.129.2.1 OpenALSource . . . 699
10.129.2.2 ~OpenALSource . . . 700
10.129.3 Member Function Documentation . . . 700
10.129.3.1 FillBufferFromFile . . . 700
10.129.3.2 FillBufferFromPCM . . . 700
10.129.3.3 GetCollisionNames . . . 700

10.129.3.4 GetOnContact . . . 700
10.129.3.5 HasCollisionName . . . 701
10.129.3.6 IsPlaying . . . 701
10.129.3.7 Load . . . 701
10.129.3.8 Pause . . . 701
10.129.3.9 Play . . . 701
10.129.3.10 Rewind . . . 701
10.129.3.11 SetGain . . . 701
10.129.3.12 SetLoop . . . 702
10.129.3.13 SetPitch . . . 702
10.129.3.14 SetPose . . . 702
10.129.3.15 SetVelocity . . . 702
10.129.3.16 Stop . . . 703
10.130 gazebo::rendering::OrbitViewController Class Reference . . . 703
10.130.1 Detailed Description . . . 704
10.130.2 Constructor & Destructor Documentation . . . 704
10.130.2.1 OrbitViewController . . . 704
10.130.2.2 ~OrbitViewController . . . 704
10.130.3 Member Function Documentation . . . 704
10.130.3.1 GetFocalPoint . . . 704
10.130.3.2 GetTypeString . . . 705
10.130.3.3 HandleKeyPressEvent . . . 705
10.130.3.4 HandleKeyReleaseEvent . . . 705
10.130.3.5 HandleMouseEvent . . . 705
10.130.3.6 Init . . . 705
10.130.3.7 Init . . . 705
10.130.3.8 SetDistance . . . 706
10.130.3.9 SetFocalPoint . . . 706
10.130.3.10 Update . . . 706
10.131 gazebo::common::ParamT< T > Class Template Reference . . . 706
10.132 gazebo::physics::PhysicsEngine Class Reference . . . 706
10.132.1 Detailed Description . . . 710
10.132.2 Constructor & Destructor Documentation . . . 710
10.132.2.1 PhysicsEngine . . . 710
10.132.2.2 ~PhysicsEngine . . . 710
10.132.3 Member Function Documentation . . . 710
10.132.3.1 CreateCollision . . . 710

10.132.3.2 CreateCollision . . . 710
10.132.3.3 CreateJoint . . . 711
10.132.3.4 CreateLink . . . 711
10.132.3.5 CreateModel . . . 711
10.132.3.6 CreateShape . . . 711
10.132.3.7 DebugPrint . . . 712
10.132.3.8 Fini . . . 712
10.132.3.9 GetAutoDisableFlag . . . 712
10.132.3.10 GetContactManager . . . 712
10.132.3.11 GetContactMaxCorrectingVel . . . 712
10.132.3.12 GetContactSurfaceLayer . . . 712
10.132.3.13 GetGravity . . . 713
10.132.3.14 GetMaxContacts . . . 713
10.132.3.15 GetMaxStepSize . . . 713
10.132.3.16 GetParam . . . 713
10.132.3.17 GetPhysicsUpdateMutex . . . 713
10.132.3.18 GetRealTimeUpdateRate . . . 713
10.132.3.19 GetSORPGSIters . . . 714
10.132.3.20 GetSORPGSPreconIters . . . 714
10.132.3.21 GetSORPGSW . . . 714
10.132.3.22 GetTargetRealTimeFactor . . . 714
10.132.3.23 GetType . . . 714
10.132.3.24 GetUpdatePeriod . . . 715
10.132.3.25 GetWorldCFM . . . 715
10.132.3.26 GetWorldERP . . . 715
10.132.3.27 Init . . . 715
10.132.3.28 InitForThread . . . 715
10.132.3.29 Load . . . 715
10.132.3.30 OnPhysicsMsg . . . 716
10.132.3.31 OnRequest . . . 716
10.132.3.32 Reset . . . 716
10.132.3.33 SetAutoDisableFlag . . . 716
10.132.3.34 SetContactMaxCorrectingVel . . . 716
10.132.3.35 SetContactSurfaceLayer . . . 716
10.132.3.36 SetGravity . . . 717
10.132.3.37 SetMaxContacts . . . 717
10.132.3.38 SetMaxStepSize . . . 717

10.132.3.39 SetParam . . . 717
10.132.3.40 SetRealTimeUpdateRate . . . 717
10.132.3.41 SetSeed . . . 718
10.132.3.42 SetSORPGSIters . . . 718
10.132.3.43 SetSORPGSPreconIters . . . 718
10.132.3.44 SetSORPGSW . . . 718
10.132.3.45 SetTargetRealTimeFactor . . . 718
10.132.3.46 SetWorldCFM . . . 719
10.132.3.47 SetWorldERP . . . 719
10.132.3.48 UpdateCollision . . . 719
10.132.3.49 UpdatePhysics . . . 719
10.132.4 Member Data Documentation . . . 719
10.132.4.1 contactManager . . . 719
10.132.4.2 maxStepSize . . . 719
10.132.4.3 node . . . 719
10.132.4.4 physicsSub . . . 720
10.132.4.5 physicsUpdateMutex . . . 720
10.132.4.6 realTimeUpdateRate . . . 720
10.132.4.7 requestSub . . . 720
10.132.4.8 responsePub . . . 720
10.132.4.9 sdf . . . 720
10.132.4.10 targetRealTimeFactor . . . 720
10.132.4.11 world . . . 720
10.133 gazebo::physics::PhysicsFactory Class Reference . . . 720
10.133.1 Detailed Description . . . 721
10.133.2 Member Function Documentation . . . 721
10.133.2.1 IsRegistered . . . 721
10.133.2.2 NewPhysicsEngine . . . 721
10.133.2.3 RegisterAll . . . 721
10.133.2.4 RegisterPhysicsEngine . . . 721
10.134 gazebo::common::PID Class Reference . . . 722
10.134.1 Detailed Description . . . 723
10.134.2 Constructor & Destructor Documentation . . . 723
10.134.2.1 PID . . . 723
10.134.2.2 ~PID . . . 723
10.134.3 Member Function Documentation . . . 723
10.134.3.1 GetCmd . . . 723

10.134.3.2 GetErrors . . . 723
10.134.3.3 Init . . . 724
10.134.3.4 operator= . . . 724
10.134.3.5 Reset . . . 724
10.134.3.6 SetCmd . . . 724
10.134.3.7 SetCmdMax . . . 724
10.134.3.8 SetCmdMin . . . 725
10.134.3.9 SetDGain . . . 725
10.134.3.10 SetIGain . . . 725
10.134.3.11 SetIMax . . . 725
10.134.3.12 SetIMin . . . 725
10.134.3.13 SetPGain . . . 725
10.134.3.14 Update . . . 726
10.135 gazebo::math::Plane Class Reference . . . 726
10.135.1 Detailed Description . . . 727
10.135.2 Constructor & Destructor Documentation . . . 727
10.135.2.1 Plane . . . 727
10.135.2.2 Plane . . . 727
10.135.2.3 Plane . . . 727
10.135.2.4 ~Plane . . . 727
10.135.3 Member Function Documentation . . . 727
10.135.3.1 Distance . . . 727
10.135.3.2 operator= . . . 728
10.135.3.3 Set . . . 728
10.135.4 Member Data Documentation . . . 728
10.135.4.1 d . . . 728
10.135.4.2 normal . . . 728
10.135.4.3 size . . . 728
10.136 gazebo::physics::PlaneShape Class Reference . . . 728
10.136.1 Detailed Description . . . 730
10.136.2 Constructor & Destructor Documentation . . . 730
10.136.2.1 PlaneShape . . . 730
10.136.2.2 ~PlaneShape . . . 730
10.136.3 Member Function Documentation . . . 730
10.136.3.1 CreatePlane . . . 730
10.136.3.2 FillMsg . . . 730
10.136.3.3 GetNormal . . . 731

10.136.3.4 GetSize . . . 731
10.136.3.5 Init . . . 731
10.136.3.6 ProcessMsg . . . 731
10.136.3.7 SetAltitude . . . 731
10.136.3.8 SetNormal . . . 731
10.136.3.9 SetScale . . . 732
10.136.3.10 SetSize . . . 732
10.137 gazebo::PluginT< T > Class Template Reference . . . 732
10.137.1 Detailed Description . . . 733
10.137.2 Member Typedef Documentation . . . 733
10.137.2.1 TPtr . . . 733
10.137.3 Constructor & Destructor Documentation . . . 733
10.137.3.1 PluginT . . . 733
10.137.3.2 ~PluginT . . . 733
10.137.4 Member Function Documentation . . . 733
10.137.4.1 Create . . . 733
10.137.4.2 GetFilename . . . 734
10.137.4.3 GetHandle . . . 734
10.137.4.4 GetType . . . 734
10.137.5 Member Data Documentation . . . 734
10.137.5.1 filename . . . 734
10.137.5.2 handle . . . 734
10.137.5.3 type . . . 734
10.138 gazebo::math::Pose Class Reference . . . 734
10.138.1 Detailed Description . . . 736
10.138.2 Constructor & Destructor Documentation . . . 736
10.138.2.1 Pose . . . 736
10.138.2.2 Pose . . . 737
10.138.2.3 Pose . . . 737
10.138.2.4 Pose . . . 737
10.138.2.5 ~Pose . . . 737
10.138.3 Member Function Documentation . . . 737
10.138.3.1 CoordPoseSolve . . . 737
10.138.3.2 CoordPositionAdd . . . 737
10.138.3.3 CoordPositionAdd . . . 738
10.138.3.4 CoordPositionSub . . . 738
10.138.3.5 CoordRotationAdd . . . 738

10.138.3.6 CoordRotationSub . . . 739
10.138.3.7 Correct . . . 739
10.138.3.8 GetInverse . . . 739
10.138.3.9 IsFinite . . . 739
10.138.3.10 operator!= . . . 739
10.138.3.11 operator* . . . 739
10.138.3.12 operator+ . . . 740
10.138.3.13 operator+= . . . 740
10.138.3.14 operator- . . . 740
10.138.3.15 operator- . . . 740
10.138.3.16 operator-= . . . 741
10.138.3.17 operator= . . . 741
10.138.3.18 operator== . . . 741
10.138.3.19 Reset . . . 741
10.138.3.20 RotatePositionAboutOrigin . . . 741
10.138.3.21 Round . . . 742
10.138.3.22 Set . . . 742
10.138.3.23 Set . . . 742
10.138.3.24 Set . . . 742
10.138.4 Friends And Related Function Documentation . . . 742
10.138.4.1 operator<< . . . 742
10.138.4.2 operator>> . . . 743
10.138.5 Member Data Documentation . . . 743
10.138.5.1 pos . . . 743
10.138.5.2 rot . . . 743
10.138.5.3 Zero . . . 743
10.139 gazebo::common::PoseAnimation Class Reference . . . 743
10.139.1 Detailed Description . . . 744
10.139.2 Constructor & Destructor Documentation . . . 744
10.139.2.1 PoseAnimation . . . 744
10.139.2.2 ~PoseAnimation . . . 745
10.139.3 Member Function Documentation . . . 745
10.139.3.1 BuildInterpolationSplines . . . 745
10.139.3.2 CreateKeyFrame . . . 745
10.139.3.3 GetInterpolatedKeyFrame . . . 745
10.139.3.4 GetInterpolatedKeyFrame . . . 745
10.140 gazebo::common::PoseKeyFrame Class Reference . . . 746

10.140.1 Detailed Description . . . 746
10.140.2 Constructor & Destructor Documentation . . . 747
10.140.2.1 PoseKeyFrame . . . 747
10.140.2.2 ~PoseKeyFrame . . . 747
10.140.3 Member Function Documentation . . . 747
10.140.3.1 GetRotation . . . 747
10.140.3.2 GetTranslation . . . 747
10.140.3.3 SetRotation . . . 747
10.140.3.4 SetTranslation . . . 747
10.140.4 Member Data Documentation . . . 748
10.140.4.1 rotate . . . 748
10.140.4.2 translate . . . 748
10.141 gazebo::rendering::Projector Class Reference . . . 748
10.141.1 Detailed Description . . . 748
10.141.2 Constructor & Destructor Documentation . . . 749
10.141.2.1 Projector . . . 749
10.141.2.2 ~Projector . . . 749
10.141.3 Member Function Documentation . . . 749
10.141.3.1 GetParent . . . 749
10.141.3.2 Load . . . 749
10.141.3.3 Load . . . 749
10.141.3.4 Load . . . 749
10.141.3.5 SetEnabled . . . 750
10.141.3.6 SetTexture . . . 750
10.141.3.7 Toggle . . . 750
10.142 gazebo::transport::Publication Class Reference . . . 750
10.142.1 Detailed Description . . . 751
10.142.2 Constructor & Destructor Documentation . . . 751
10.142.2.1 Publication . . . 751
10.142.2.2 ~Publication . . . 752
10.142.3 Member Function Documentation . . . 752
10.142.3.1 AddPublisher . . . 752
10.142.3.2 AddSubscription . . . 752
10.142.3.3 AddSubscription . . . 752
10.142.3.4 AddTransport . . . 752
10.142.3.5 GetCallbackCount . . . 752
10.142.3.6 GetLocallyAdvertised . . . 753

10.142.3.7 GetMsgType . . . 753
10.142.3.8 GetNodeCount . . . 753
10.142.3.9 GetRemoteSubscriptionCount . . . 753
10.142.3.10 GetTransportCount . . . 753
10.142.3.11 HasTransport . . . 753
10.142.3.12 LocalPublish . . . 754
10.142.3.13 Publish . . . 754
10.142.3.14 RemoveSubscription . . . 754
10.142.3.15 RemoveSubscription . . . 754
10.142.3.16 RemoveTransport . . . 754
10.142.3.17 SetLocallyAdvertised . . . 755
10.143 gazebo::transport::PublicationTransport Class Reference . . . 755
10.143.1 Detailed Description . . . 755
10.143.2 Constructor & Destructor Documentation . . . 755
10.143.2.1 PublicationTransport . . . 755
10.143.2.2 ~PublicationTransport . . . 756
10.143.3 Member Function Documentation . . . 756
10.143.3.1 AddCallback . . . 756
10.143.3.2 Fini . . . 756
10.143.3.3 GetConnection . . . 756
10.143.3.4 GetMsgType . . . 756
10.143.3.5 GetTopic . . . 756
10.143.3.6 Init . . . 757
10.144 gazebo::transport::Publisher Class Reference . . . 757
10.144.1 Detailed Description . . . 758
10.144.2 Constructor & Destructor Documentation . . . 758
10.144.2.1 Publisher . . . 758
10.144.2.2 ~Publisher . . . 758
10.144.3 Member Function Documentation . . . 758
10.144.3.1 GetMsgType . . . 758
10.144.3.2 GetOutgoingCount . . . 758
10.144.3.3 GetPrevMsg . . . 758
10.144.3.4 GetPrevMsgPtr . . . 759
10.144.3.5 GetTopic . . . 759
10.144.3.6 HasConnections . . . 759
10.144.3.7 Publish . . . 759
10.144.3.8 Publish . . . 759

10.144.3.9 SendMessage . . . 760
10.144.3.10 SetNode . . . 760
10.144.3.11 SetPublication . . . 760
10.144.3.12 WaitForConnection . . . 760
10.144.3.13 WaitForConnection . . . 760
10.145 QuadNode Class Reference . . . 760
10.146 gazebo::math::Quaternion Class Reference . . . 761
10.146.1 Detailed Description . . . 763
10.146.2 Constructor & Destructor Documentation . . . 764
10.146.2.1 Quaternion . . . 764
10.146.2.2 Quaternion . . . 764
10.146.2.3 Quaternion . . . 764
10.146.2.4 Quaternion . . . 764
10.146.2.5 Quaternion . . . 764
10.146.2.6 Quaternion . . . 764
10.146.2.7 ~Quaternion . . . 765
10.146.3 Member Function Documentation . . . 765
10.146.3.1 Correct . . . 765
10.146.3.2 Dot . . . 765
10.146.3.3 EulerToQuaternion . . . 765
10.146.3.4 EulerToQuaternion . . . 765
10.146.3.5 GetAsAxis . . . 766
10.146.3.6 GetAsEuler . . . 766
10.146.3.7 GetAsMatrix3 . . . 766
10.146.3.8 GetAsMatrix4 . . . 766
10.146.3.9 GetExp . . . 766
10.146.3.10 GetInverse . . . 766
10.146.3.11 GetLog . . . 767
10.146.3.12 GetPitch . . . 767
10.146.3.13 GetRoll . . . 767
10.146.3.14 GetXAxis . . . 767
10.146.3.15 GetYaw . . . 767
10.146.3.16 GetYAxis . . . 767
10.146.3.17 GetZAxis . . . 768
10.146.3.18 Invert . . . 768
10.146.3.19 IsFinite . . . 768
10.146.3.20 Normalize . . . 768

10.146.3.21 operator!= . . . 768
10.146.3.22 operator* . . . 768
10.146.3.23 operator* . . . 769
10.146.3.24 operator* . . . 769
10.146.3.25 operator*= . . . 769
10.146.3.26 operator+ . . . 769
10.146.3.27 operator+= . . . 770
10.146.3.28 operator- . . . 770
10.146.3.29 operator- . . . 770
10.146.3.30 operator-= . . . 770
10.146.3.31 operator= . . . 770
10.146.3.32 operator== . . . 771
10.146.3.33 RotateVector . . . 771
10.146.3.34 RotateVectorReverse . . . 771
10.146.3.35 Round . . . 771
10.146.3.36 Scale . . . 772
10.146.3.37 Set . . . 772
10.146.3.38 SetFromAxis . . . 772
10.146.3.39 SetFromAxis . . . 772
10.146.3.40 SetFromEuler . . . 772
10.146.3.41 SetFromEuler . . . 773
10.146.3.42 SetToIdentity . . . 773
10.146.3.43 Slerp . . . 773
10.146.3.44 Squad . . . 773
10.146.4 Friends And Related Function Documentation . . . 773
10.146.4.1 operator<< . . . 773
10.146.4.2 operator>> . . . 774
10.146.5 Member Data Documentation . . . 774
10.146.5.1 w . . . 774
10.146.5.2 x . . . 774
10.146.5.3 y . . . 774
10.146.5.4 z . . . 774
10.147 gazebo::math::Rand Class Reference . . . 775
10.147.1 Detailed Description . . . 775
10.147.2 Member Function Documentation . . . 775
10.147.2.1 GetDblNormal . . . 775
10.147.2.2 GetDblUniform . . . 775

10.147.2.3 GetIntNormal . . . 775
10.147.2.4 GetIntUniform . . . 776
10.147.2.5 GetSeed . . . 776
10.147.2.6 SetSeed . . . 776
10.148 gazebo::transport::RawCallbackHelper Class Reference . . . 776
10.148.1 Detailed Description . . . 777
10.148.2 Constructor & Destructor Documentation . . . 777
10.148.2.1 RawCallbackHelper . . . 777
10.148.3 Member Function Documentation . . . 778
10.148.3.1 GetMsgType . . . 778
10.148.3.2 HandleData . . . 778
10.148.3.3 HandleMessage . . . 778
10.148.3.4 IsLocal . . . 778
10.149 gazebo::sensors::RaySensor Class Reference . . . 779
10.149.1 Detailed Description . . . 781
10.149.2 Constructor & Destructor Documentation . . . 781
10.149.2.1 RaySensor . . . 781
10.149.2.2 ~RaySensor . . . 781
10.149.3 Member Function Documentation . . . 781
10.149.3.1 Fini . . . 781
10.149.3.2 GetAngleMax . . . 781
10.149.3.3 GetAngleMin . . . 781
10.149.3.4 GetAngleResolution . . . 781
10.149.3.5 GetFiducial . . . 782
10.149.3.6 GetLaserShape . . . 782
10.149.3.7 GetRange . . . 782
10.149.3.8 GetRangeCount . . . 782
10.149.3.9 GetRangeMax . . . 783
10.149.3.10 GetRangeMin . . . 783
10.149.3.11 GetRangeResolution . . . 783
10.149.3.12 GetRanges . . . 783
10.149.3.13 GetRayCount . . . 783
10.149.3.14 GetRetro . . . 783
10.149.3.15 GetTopic . . . 784
10.149.3.16 GetVerticalAngleMax . . . 784
10.149.3.17 GetVerticalAngleMin . . . 784
10.149.3.18 GetVerticalRangeCount . . . 784

10.149.3.19 GetVerticalRayCount . . . 785
10.149.3.20 Init . . . 785
10.149.3.21 IsActive . . . 785
10.149.3.22 Load . . . 785
10.149.3.23 UpdateImpl . . . 785
10.150 gazebo::physics::RayShape Class Reference . . . 786
10.150.1 Detailed Description . . . 787
10.150.2 Constructor & Destructor Documentation . . . 788
10.150.2.1 RayShape . . . 788
10.150.2.2 RayShape . . . 788
10.150.2.3 ~RayShape . . . 788
10.150.3 Member Function Documentation . . . 788
10.150.3.1 FillMsg . . . 788
10.150.3.2 GetFiducial . . . 788
10.150.3.3 GetGlobalPoints . . . 788
10.150.3.4 GetIntersection . . . 789
10.150.3.5 GetLength . . . 789
10.150.3.6 GetRelativePoints . . . 789
10.150.3.7 GetRetro . . . 789
10.150.3.8 Init . . . 789
10.150.3.9 ProcessMsg . . . 789
10.150.3.10 SetFiducial . . . 790
10.150.3.11 SetLength . . . 790
10.150.3.12 SetPoints . . . 790
10.150.3.13 SetRetro . . . 790
10.150.3.14 SetScale . . . 790
10.150.3.15 Update . . . 791
10.150.4 Member Data Documentation . . . 791
10.150.4.1 contactFiducial . . . 791
10.150.4.2 contactLen . . . 791
10.150.4.3 contactRetro . . . 791
10.150.4.4 globalEndPos . . . 791
10.150.4.5 globalStartPos . . . 791
10.150.4.6 relativeEndPos . . . 791
10.150.4.7 relativeStartPos . . . 791
10.151 gazebo::rendering::RenderEngine Class Reference . . . 791
10.151.1 Detailed Description . . . 793

10.151.2 Member Enumeration Documentation . . . 793
10.151.2.1 RenderPathType . . . 793
10.151.3 Member Function Documentation . . . 793
10.151.3.1 AddResourcePath . . . 793
10.151.3.2 CreateScene . . . 794
10.151.3.3 Fini . . . 794
10.151.3.4 GetRenderPathType . . . 794
10.151.3.5 GetScene . . . 794
10.151.3.6 GetScene . . . 794
10.151.3.7 GetSceneCount . . . 795
10.151.3.8 GetWindowManager . . . 795
10.151.3.9 Init . . . 795
10.151.3.10 Load . . . 795
10.151.3.11 RemoveScene . . . 795
10.151.4 Member Data Documentation . . . 795
10.151.4.1 dummyContext . . . 795
10.151.4.2 dummyDisplay . . . 795
10.151.4.3 dummyWindowId . . . 795
10.151.4.4 root . . . 796
10.152 gazebo::sensors::RFIDSensor Class Reference . . . 796
10.152.1 Detailed Description . . . 797
10.152.2 Constructor & Destructor Documentation . . . 797
10.152.2.1 RFIDSensor . . . 797
10.152.2.2 ~RFIDSensor . . . 797
10.152.3 Member Function Documentation . . . 797
10.152.3.1 AddTag . . . 797
10.152.3.2 Fini . . . 797
10.152.3.3 Init . . . 797
10.152.3.4 Load . . . 797
10.152.3.5 Load . . . 798
10.152.3.6 UpdateImpl . . . 798
10.153 gazebo::sensors::RFIDTag Class Reference . . . 798
10.153.1 Detailed Description . . . 799
10.153.2 Constructor & Destructor Documentation . . . 800
10.153.2.1 RFIDTag . . . 800
10.153.2.2 ~RFIDTag . . . 800
10.153.3 Member Function Documentation . . . 800

10.153.3.1Fini . . . 800
10.153.3.2GetTagPose . . . 800
10.153.3.3Init . . . 800
10.153.3.4Load . . . 800
10.153.3.5Load . . . 800
10.153.3.6UpdateImpl . . . 800
10.154gazebo::rendering::RFIDTagVisual Class Reference . . . 801
10.154.1Detailed Description . . . 802
10.154.2Constructor & Destructor Documentation . . . 802
10.154.2.1RFIDTagVisual . . . 802
10.154.2.2~RFIDTagVisual . . . 802
10.155gazebo::rendering::RFIDVisual Class Reference . . . 802
10.155.1Detailed Description . . . 803
10.155.2Constructor & Destructor Documentation . . . 803
10.155.2.1RFIDVisual . . . 803
10.155.2.2~RFIDVisual . . . 804
10.156Road Class Reference . . . 804
10.156.1Detailed Description . . . 804
10.157gazebo::physics::Road Class Reference . . . 804
10.157.1Detailed Description . . . 805
10.157.2Constructor & Destructor Documentation . . . 805
10.157.2.1Road . . . 805
10.157.2.2~Road . . . 805
10.157.3Member Function Documentation . . . 805
10.157.3.1Init . . . 805
10.157.3.2Load . . . 805
10.158gazebo::rendering::Road2d Class Reference . . . 806
10.158.1Constructor & Destructor Documentation . . . 806
10.158.1.1Road2d . . . 806
10.158.1.2~Road2d . . . 806
10.158.2Member Function Documentation . . . 806
10.158.2.1Load . . . 806
10.159gazebo::math::RotationSpline Class Reference . . . 806
10.159.1Detailed Description . . . 807
10.159.2Constructor & Destructor Documentation . . . 807
10.159.2.1RotationSpline . . . 807
10.159.2.2~RotationSpline . . . 807

10.159.3 Member Function Documentation . . . 808
10.159.3.1 AddPoint . . . 808
10.159.3.2 Clear . . . 808
10.159.3.3 GetNumPoints . . . 808
10.159.3.4 GetPoint . . . 808
10.159.3.5 Interpolate . . . 808
10.159.3.6 Interpolate . . . 809
10.159.3.7 RecalcTangents . . . 809
10.159.3.8 SetAutoCalculate . . . 809
10.159.3.9 UpdatePoint . . . 809
10.159.4 Member Data Documentation . . . 810
10.159.4.1 autoCalc . . . 810
10.159.4.2 points . . . 810
10.159.4.3 tangents . . . 810
10.160 gazebo::rendering::RTShaderSystem Class Reference . . . 810
10.160.1 Detailed Description . . . 811
10.160.2 Member Enumeration Documentation . . . 812
10.160.2.1 LightingModel . . . 812
10.160.3 Member Function Documentation . . . 812
10.160.3.1 AddScene . . . 812
10.160.3.2 ApplyShadows . . . 812
10.160.3.3 AttachEntity . . . 812
10.160.3.4 AttachViewport . . . 812
10.160.3.5 Clear . . . 813
10.160.3.6 DetachEntity . . . 813
10.160.3.7 DetachViewport . . . 813
10.160.3.8 Fini . . . 813
10.160.3.9 GenerateShaders . . . 813
10.160.3.10 GetPSSMShadowCameraSetup . . . 813
10.160.3.11 Init . . . 813
10.160.3.12 RemoveScene . . . 814
10.160.3.13 RemoveShadows . . . 814
10.160.3.14 SetPerPixelLighting . . . 814
10.160.3.15 UpdateShaders . . . 814
10.161 gazebo::rendering::Scene Class Reference . . . 814
10.161.1 Detailed Description . . . 818
10.161.2 Member Enumeration Documentation . . . 818

10.161.2.1 SkyXMode . . . 818
10.161.3 Constructor & Destructor Documentation . . . 819
10.161.3.1 Scene . . . 819
10.161.3.2 ~Scene . . . 819
10.161.4 Member Function Documentation . . . 819
10.161.4.1 AddVisual . . . 819
10.161.4.2 Clear . . . 819
10.161.4.3 CloneVisual . . . 819
10.161.4.4 CreateCamera . . . 819
10.161.4.5 CreateDepthCamera . . . 820
10.161.4.6 CreateGpuLaser . . . 820
10.161.4.7 CreateGrid . . . 820
10.161.4.8 CreateUserCamera . . . 820
10.161.4.9 DrawLine . . . 821
10.161.4.10 GetAmbientColor . . . 821
10.161.4.11 GetBackgroundColor . . . 821
10.161.4.12 GetCamera . . . 821
10.161.4.13 GetCamera . . . 821
10.161.4.14 GetCameraCount . . . 822
10.161.4.15 GetFirstContact . . . 822
10.161.4.16 GetGrid . . . 822
10.161.4.17 GetGridCount . . . 822
10.161.4.18 GetHeightBelowPoint . . . 823
10.161.4.19 GetHeightmap . . . 823
10.161.4.20 GetId . . . 823
10.161.4.21 GetIdString . . . 823
10.161.4.22 GetInitialized . . . 823
10.161.4.23 GetLight . . . 823
10.161.4.24 GetLight . . . 824
10.161.4.25 GetLightCount . . . 824
10.161.4.26 GetManager . . . 824
10.161.4.27 GetModelVisualAt . . . 824
10.161.4.28 GetName . . . 825
10.161.4.29 GetSelectedVisual . . . 825
10.161.4.30 GetShadowsEnabled . . . 825
10.161.4.31 GetShowClouds . . . 825
10.161.4.32 GetSimTime . . . 825

10.161.4.33 GetUserCamera . . . 825
10.161.4.34 GetUserCameraCount . . . 826
10.161.4.35 GetVisual . . . 826
10.161.4.36 GetVisual . . . 826
10.161.4.37 GetVisualAt . . . 826
10.161.4.38 GetVisualAt . . . 827
10.161.4.39 GetVisualBelow . . . 827
10.161.4.40 GetVisualCount . . . 827
10.161.4.41 GetVisualsBelowPoint . . . 827
10.161.4.42 GetWorldVisual . . . 828
10.161.4.43 Init . . . 828
10.161.4.44 Load . . . 828
10.161.4.45 Load . . . 828
10.161.4.46 PreRender . . . 828
10.161.4.47 PrintSceneGraph . . . 828
10.161.4.48 RemoveCamera . . . 828
10.161.4.49 RemoveProjectors . . . 828
10.161.4.50 RemoveVisual . . . 828
10.161.4.51 SelectVisual . . . 829
10.161.4.52 SetAmbientColor . . . 829
10.161.4.53 SetBackgroundColor . . . 829
10.161.4.54 SetFog . . . 829
10.161.4.55 SetGrid . . . 829
10.161.4.56 SetShadowsEnabled . . . 830
10.161.4.57 SetSkyXMode . . . 830
10.161.4.58 SetTransparent . . . 830
10.161.4.59 SetVisible . . . 830
10.161.4.60 SetWireframe . . . 830
10.161.4.61 ShowClouds . . . 831
10.161.4.62 ShowCollisions . . . 831
10.161.4.63 ShowCOMs . . . 831
10.161.4.64 ShowContacts . . . 831
10.161.4.65 ShowJoints . . . 831
10.161.4.66 SnapVisualToNearestBelow . . . 831
10.161.4.67 StripSceneName . . . 832
10.161.5 Member Data Documentation . . . 832
10.161.5.1 skyx . . . 832

10.162 gazebo::physics::ScrewJoint< T > Class Template Reference . . . 832
10.162.1 Detailed Description . . . 833
10.162.2 Constructor & Destructor Documentation . . . 833
10.162.2.1 ScrewJoint . . . 833
10.162.2.2 ~ScrewJoint . . . 833
10.162.3 Member Function Documentation . . . 833
10.162.3.1 GetAnchor . . . 834
10.162.3.2 GetAngleCount . . . 834
10.162.3.3 GetThreadPitch . . . 834
10.162.3.4 Load . . . 834
10.162.3.5 SetAnchor . . . 834
10.162.3.6 SetThreadPitch . . . 835
10.162.4 Member Data Documentation . . . 835
10.162.4.1 fakeAnchor . . . 835
10.162.4.2 threadPitch . . . 835
10.163 gazebo::rendering::SelectionObj Class Reference . . . 835
10.163.1 Detailed Description . . . 837
10.164 gazebo::sensors::Sensor Class Reference . . . 837
10.164.1 Detailed Description . . . 840
10.164.2 Constructor & Destructor Documentation . . . 840
10.164.2.1 Sensor . . . 840
10.164.2.2 ~Sensor . . . 841
10.164.3 Member Function Documentation . . . 841
10.164.3.1 ConnectUpdated . . . 841
10.164.3.2 DisconnectUpdated . . . 841
10.164.3.3 FillMsg . . . 841
10.164.3.4 Fini . . . 841
10.164.3.5 GetCategory . . . 842
10.164.3.6 GetId . . . 842
10.164.3.7 GetLastMeasurementTime . . . 842
10.164.3.8 GetLastUpdateTime . . . 842
10.164.3.9 GetName . . . 842
10.164.3.10 GetParentId . . . 843
10.164.3.11 GetParentName . . . 843
10.164.3.12 GetPose . . . 843
10.164.3.13 GetScopedName . . . 843
10.164.3.14 GetTopic . . . 843

10.164.3.16 GetType . . . 844
10.164.3.17 GetUpdateRate . . . 844
10.164.3.18 GetVisualize . . . 844
10.164.3.19 GetWorldName . . . 844
10.164.3.20 Init . . . 844
10.164.3.21 IsActive . . . 844
10.164.3.22 Load . . . 845
10.164.3.23 Load . . . 845
10.164.3.24 ResetLastUpdateTime . . . 845
10.164.3.25 SetActive . . . 845
10.164.3.26 SetParent . . . 846
10.164.3.27 SetParent . . . 846
10.164.3.28 SetUpdateRate . . . 846
10.164.3.29 Update . . . 846
10.164.3.30 UpdateImpl . . . 846
10.164.4 Member Data Documentation . . . 847
10.164.4.1 active . . . 847
10.164.4.2 connections . . . 847
10.164.4.3 lastMeasurementTime . . . 847
10.164.4.4 lastUpdateTime . . . 847
10.164.4.5 mutexLastUpdateTime . . . 847
10.164.4.6 node . . . 847
10.164.4.7 parentId . . . 847
10.164.4.8 parentName . . . 847
10.164.4.9 plugins . . . 847
10.164.4.10 pose . . . 847
10.164.4.11 poseSub . . . 848
10.164.4.12 scene . . . 848
10.164.4.13 sdf . . . 848
10.164.4.14 updatePeriod . . . 848
10.164.4.15 world . . . 848
10.165 SensorFactor Class Reference . . . 848
10.165.1 Detailed Description . . . 848
10.166 gazebo::sensors::SensorFactory Class Reference . . . 848
10.166.1 Member Function Documentation . . . 849
10.166.1.1 GetSensorTypes . . . 849
10.166.1.2 NewSensor . . . 849

10.166.1.3 RegisterAll . . . 849
10.166.1.4 RegisterSensor . . . 850
10.167 gazebo::sensors::SensorManager Class Reference . . . 850
10.167.1 Detailed Description . . . 851
10.167.2 Member Function Documentation . . . 851
10.167.2.1 CreateSensor . . . 851
10.167.2.2 CreateSensor . . . 851
10.167.2.3 Fini . . . 852
10.167.2.4 GetSensor . . . 852
10.167.2.5 GetSensors . . . 852
10.167.2.6 GetSensorTypes . . . 852
10.167.2.7 Init . . . 852
10.167.2.8 RemoveSensor . . . 852
10.167.2.9 RemoveSensors . . . 852
10.167.2.10 ResetLastUpdateTimes . . . 853
10.167.2.11 RunThreads . . . 853
10.167.2.12 SensorsInitialized . . . 853
10.167.2.13 Stop . . . 853
10.167.2.14 Update . . . 853
10.168 gazebo::SensorPlugin Class Reference . . . 853
10.168.1 Detailed Description . . . 854
10.168.2 Constructor & Destructor Documentation . . . 854
10.168.2.1 SensorPlugin . . . 854
10.168.2.2 ~SensorPlugin . . . 854
10.168.3 Member Function Documentation . . . 855
10.168.3.1 Init . . . 855
10.168.3.2 Load . . . 855
10.168.3.3 Reset . . . 855
10.169 gazebo::Server Class Reference . . . 855
10.169.1 Constructor & Destructor Documentation . . . 856
10.169.1.1 Server . . . 856
10.169.1.2 ~Server . . . 856
10.169.2 Member Function Documentation . . . 856
10.169.2.1 Fini . . . 856
10.169.2.2 GetInitialized . . . 856
10.169.2.3 Init . . . 856
10.169.2.4 LoadFile . . . 856

10.169.2.5 LoadString . . . 856
10.169.2.6 ParseArgs . . . 856
10.169.2.7 PreLoad . . . 856
10.169.2.8 PrintUsage . . . 856
10.169.2.9 Run . . . 856
10.169.2.10 SetParams . . . 856
10.169.2.11 Stop . . . 856
10.169.3 Member Data Documentation . . . 856
10.169.3.1 systemPluginsArgc . . . 857
10.169.3.2 systemPluginsArgv . . . 857
10.170 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference . . . 857
10.170.1 Detailed Description . . . 858
10.170.2 Member Function Documentation . . . 858
10.170.2.1 defaultVpParams . . . 858
10.170.2.2 generateFragmentProgram . . . 858
10.170.2.3 generateVertexProgram . . . 858
10.170.2.4 generateVertexProgramSource . . . 858
10.170.2.5 generateVpDynamicShadows . . . 858
10.170.2.6 generateVpDynamicShadowsParams . . . 858
10.170.2.7 generateVpFooter . . . 858
10.170.2.8 generateVpHeader . . . 858
10.171 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference . . . 859
10.171.1 Detailed Description . . . 860
10.171.2 Member Function Documentation . . . 860
10.171.2.1 defaultVpParams . . . 860
10.171.2.2 generateFpDynamicShadows . . . 860
10.171.2.3 generateFpDynamicShadowsHelpers . . . 860
10.171.2.4 generateFpDynamicShadowsParams . . . 860
10.171.2.5 generateFpFooter . . . 860
10.171.2.6 generateFpHeader . . . 860
10.171.2.7 generateFpLayer . . . 860
10.171.2.8 generateFragmentProgram . . . 861
10.171.2.9 generateFragmentProgramSource . . . 861
10.171.2.10 generateVertexProgram . . . 861
10.171.2.11 generateVertexProgramSource . . . 861
10.171.2.12 generateVpDynamicShadows . . . 861
10.171.2.13 generateVpDynamicShadowsParams . . . 861

10.171.2.14 GenerateVpFooter . . . 861
10.171.2.15 GenerateVpHeader . . . 861
10.171.2.16 UpdateParams . . . 861
10.171.2.17 UpdateVpParams . . . 861
10.172 gazebo::physics::Shape Class Reference . . . 861
10.172.1 Detailed Description . . . 863
10.172.2 Constructor & Destructor Documentation . . . 863
10.172.2.1 Shape . . . 863
10.172.2.2 ~Shape . . . 863
10.172.3 Member Function Documentation . . . 863
10.172.3.1 FillMsg . . . 863
10.172.3.2 GetScale . . . 863
10.172.3.3 Init . . . 864
10.172.3.4 ProcessMsg . . . 864
10.172.3.5 SetScale . . . 864
10.172.4 Member Data Documentation . . . 864
10.172.4.1 collisionParent . . . 864
10.172.4.2 scale . . . 865
10.173 gazebo::physics::SimbodyBallJoint Class Reference . . . 865
10.173.1 Detailed Description . . . 867
10.173.2 Constructor & Destructor Documentation . . . 867
10.173.2.1 SimbodyBallJoint . . . 867
10.173.2.2 ~SimbodyBallJoint . . . 867
10.173.3 Member Function Documentation . . . 868
10.173.3.1 GetAnchor . . . 868
10.173.3.2 GetAngleImpl . . . 868
10.173.3.3 GetAxis . . . 868
10.173.3.4 GetGlobalAxis . . . 868
10.173.3.5 GetMaxForce . . . 868
10.173.3.6 GetVelocity . . . 869
10.173.3.7 Init . . . 869
10.173.3.8 Load . . . 869
10.173.3.9 SetDamping . . . 869
10.173.3.10 SetForceImpl . . . 870
10.173.3.11 SetHighStop . . . 870
10.173.3.12 SetLowStop . . . 870
10.173.3.13 SetMaxForce . . . 870

10.173.3.13SetVelocity . . . 871
10.174gazebo::physics::SimbodyBoxShape Class Reference . . . 871
10.174.1Detailed Description . . . 872
10.174.2Constructor & Destructor Documentation . . . 872
10.174.2.1SimbodyBoxShape . . . 872
10.174.2.2~SimbodyBoxShape . . . 872
10.174.3Member Function Documentation . . . 872
10.174.3.1SetSize . . . 872
10.175gazebo::physics::SimbodyCollision Class Reference . . . 872
10.175.1Detailed Description . . . 874
10.175.2Constructor & Destructor Documentation . . . 874
10.175.2.1SimbodyCollision . . . 874
10.175.2.2~SimbodyCollision . . . 874
10.175.3Member Function Documentation . . . 874
10.175.3.1GetBoundingBox . . . 874
10.175.3.2GetCollisionShape . . . 874
10.175.3.3Load . . . 874
10.175.3.4OnPoseChange . . . 875
10.175.3.5SetCategoryBits . . . 875
10.175.3.6SetCollideBits . . . 875
10.175.3.7SetCollisionShape . . . 875
10.176gazebo::physics::SimbodyCylinderShape Class Reference . . . 875
10.176.1Detailed Description . . . 876
10.176.2Constructor & Destructor Documentation . . . 877
10.176.2.1SimbodyCylinderShape . . . 877
10.176.2.2~SimbodyCylinderShape . . . 877
10.176.3Member Function Documentation . . . 877
10.176.3.1SetSize . . . 877
10.177gazebo::physics::SimbodyHeightmapShape Class Reference . . . 877
10.177.1Detailed Description . . . 878
10.177.2Constructor & Destructor Documentation . . . 879
10.177.2.1SimbodyHeightmapShape . . . 879
10.177.2.2~SimbodyHeightmapShape . . . 879
10.177.3Member Function Documentation . . . 879
10.177.3.1Init . . . 879
10.178gazebo::physics::SimbodyHinge2Joint Class Reference . . . 879
10.178.1Detailed Description . . . 881

10.178.2 Constructor & Destructor Documentation . . . 881
10.178.2.1 SimbodyHinge2Joint . . . 881
10.178.2.2 ~SimbodyHinge2Joint . . . 881
10.178.3 Member Function Documentation . . . 882
10.178.3.1 GetAnchor . . . 882
10.178.3.2 GetAngleImpl . . . 882
10.178.3.3 GetAxis . . . 882
10.178.3.4 GetGlobalAxis . . . 882
10.178.3.5 GetHighStop . . . 882
10.178.3.6 GetLowStop . . . 883
10.178.3.7 GetMaxForce . . . 883
10.178.3.8 GetVelocity . . . 883
10.178.3.9 Init . . . 884
10.178.3.10 Load . . . 884
10.178.3.11 SetAxis . . . 884
10.178.3.12 SetDamping . . . 884
10.178.3.13 SetForceImpl . . . 884
10.178.3.14 SetHighStop . . . 885
10.178.3.15 SetLowStop . . . 885
10.178.3.16 SetMaxForce . . . 885
10.178.3.17 SetVelocity . . . 885
10.179 gazebo::physics::SimbodyHingeJoint Class Reference . . . 886
10.179.1 Detailed Description . . . 887
10.179.2 Constructor & Destructor Documentation . . . 887
10.179.2.1 SimbodyHingeJoint . . . 887
10.179.2.2 ~SimbodyHingeJoint . . . 888
10.179.3 Member Function Documentation . . . 888
10.179.3.1 GetAngleImpl . . . 888
10.179.3.2 GetGlobalAxis . . . 888
10.179.3.3 GetHighStop . . . 888
10.179.3.4 GetLowStop . . . 889
10.179.3.5 GetMaxForce . . . 889
10.179.3.6 GetVelocity . . . 889
10.179.3.7 Load . . . 889
10.179.3.8 RestoreSimbodyState . . . 890
10.179.3.9 SaveSimbodyState . . . 890
10.179.3.10 SetAxis . . . 890

10.179.3.13SetDamping . . . 890
10.179.3.14SetForceImpl . . . 890
10.179.3.15SetHighStop . . . 891
10.179.3.16SetLowStop . . . 891
10.179.3.17SetMaxForce . . . 891
10.179.3.18SetVelocity . . . 891
10.180gazebo::physics::SimbodyJoint Class Reference . . . 892
10.180.1Detailed Description . . . 894
10.180.2Constructor & Destructor Documentation . . . 894
10.180.2.1SimbodyJoint . . . 894
10.180.2.2~SimbodyJoint . . . 894
10.180.3Member Function Documentation . . . 894
10.180.3.1AreConnected . . . 894
10.180.3.2CacheForceTorque . . . 894
10.180.3.3Detach . . . 894
10.180.3.4GetAnchor . . . 894
10.180.3.5GetAttribute . . . 895
10.180.3.6GetForce . . . 895
10.180.3.7GetForceTorque . . . 895
10.180.3.8GetJointLink . . . 896
10.180.3.9GetLinkForce . . . 896
10.180.3.10GetLinkTorque . . . 896
10.180.3.11Load . . . 897
10.180.3.12Reset . . . 897
10.180.3.13RestoreSimbodyState . . . 897
10.180.3.14SaveSimbodyState . . . 897
10.180.3.15SetAnchor . . . 897
10.180.3.16SetAttribute . . . 898
10.180.3.17SetAttribute . . . 898
10.180.3.18SetAxis . . . 898
10.180.3.19SetDamping . . . 898
10.180.3.20SetForce . . . 898
10.180.3.21SetForceImpl . . . 899
10.180.4Member Data Documentation . . . 899
10.180.4.1constraint . . . 899
10.180.4.2damper . . . 899
10.180.4.3defxAB . . . 899

10.180.4.4 isReversed . . . 899
10.180.4.5 limitForce . . . 900
10.180.4.6 mobod . . . 900
10.180.4.7 mustBreakLoopHere . . . 900
10.180.4.8 physicsInitialized . . . 900
10.180.4.9 simbodyPhysics . . . 900
10.180.4.10 world . . . 900
10.180.4.11 xCB . . . 900
10.180.4.12 xPA . . . 900
10.181 gazebo::physics::SimbodyLink Class Reference . . . 900
10.181.1 Detailed Description . . . 903
10.181.2 Constructor & Destructor Documentation . . . 903
10.181.2.1 SimbodyLink . . . 903
10.181.2.2 ~SimbodyLink . . . 903
10.181.3 Member Function Documentation . . . 903
10.181.3.1 AddForce . . . 903
10.181.3.2 AddForceAtRelativePosition . . . 904
10.181.3.3 AddForceAtWorldPosition . . . 904
10.181.3.4 AddRelativeForce . . . 904
10.181.3.5 AddRelativeTorque . . . 904
10.181.3.6 AddTorque . . . 904
10.181.3.7 Fini . . . 905
10.181.3.8 GetEffectiveMassProps . . . 905
10.181.3.9 GetEnabled . . . 905
10.181.3.10 GetGravityMode . . . 905
10.181.3.11 GetMassProperties . . . 905
10.181.3.12 GetWorldAngularVel . . . 905
10.181.3.13 GetWorldCoGLinearVel . . . 905
10.181.3.14 GetWorldForce . . . 906
10.181.3.15 GetWorldLinearVel . . . 906
10.181.3.16 GetWorldLinearVel . . . 906
10.181.3.17 GetWorldTorque . . . 906
10.181.3.18 Init . . . 907
10.181.3.19 Load . . . 907
10.181.3.20 OnPoseChange . . . 907
10.181.3.21 RestoreSimbodyState . . . 907
10.181.3.22 SaveSimbodyState . . . 907

10.181.3.23 SetAngularDamping . . . 907
10.181.3.24 SetAngularVel . . . 907
10.181.3.25 SetAutoDisable . . . 907
10.181.3.26 SetDirtyPose . . . 908
10.181.3.27 SetEnabled . . . 908
10.181.3.28 SetForce . . . 908
10.181.3.29 SetGravityMode . . . 908
10.181.3.30 SetLinearDamping . . . 908
10.181.3.31 SetLinearVel . . . 908
10.181.3.32 SetLinkStatic . . . 909
10.181.3.33 SetSelfCollide . . . 909
10.181.3.34 SetTorque . . . 909
10.181.4 Member Data Documentation . . . 909
10.181.4.1 masterMobod . . . 909
10.181.4.2 mustBeBaseLink . . . 909
10.181.4.3 physicsInitialized . . . 909
10.181.4.4 slaveMobods . . . 909
10.181.4.5 slaveWelds . . . 910
10.182 gazebo::physics::SimbodyMeshShape Class Reference . . . 910
10.182.1 Detailed Description . . . 911
10.182.2 Constructor & Destructor Documentation . . . 911
10.182.2.1 SimbodyMeshShape . . . 911
10.182.2.2 ~SimbodyMeshShape . . . 911
10.182.3 Member Function Documentation . . . 911
10.182.3.1 Init . . . 911
10.182.3.2 Load . . . 911
10.183 gazebo::physics::SimbodyModel Class Reference . . . 912
10.183.1 Detailed Description . . . 913
10.183.2 Constructor & Destructor Documentation . . . 913
10.183.2.1 SimbodyModel . . . 913
10.183.2.2 ~SimbodyModel . . . 913
10.183.3 Member Function Documentation . . . 913
10.183.3.1 Init . . . 913
10.183.3.2 Load . . . 913
10.184 gazebo::physics::SimbodyMultiRayShape Class Reference . . . 913
10.184.1 Detailed Description . . . 915
10.184.2 Constructor & Destructor Documentation . . . 915

10.184.2.1 SimbodyMultiRayShape . . . 915
10.184.2.2 ~SimbodyMultiRayShape . . . 915
10.184.3 Member Function Documentation . . . 915
10.184.3.1 AddRay . . . 915
10.184.3.2 UpdateRays . . . 915
10.185 gazebo::physics::SimbodyPhysics Class Reference . . . 915
10.185.1 Detailed Description . . . 918
10.185.2 Constructor & Destructor Documentation . . . 918
10.185.2.1 SimbodyPhysics . . . 918
10.185.2.2 ~SimbodyPhysics . . . 918
10.185.3 Member Function Documentation . . . 918
10.185.3.1 CreateCollision . . . 918
10.185.3.2 CreateJoint . . . 918
10.185.3.3 CreateLink . . . 919
10.185.3.4 CreateModel . . . 919
10.185.3.5 CreateShape . . . 919
10.185.3.6 DebugPrint . . . 919
10.185.3.7 Fini . . . 919
10.185.3.8 GetDynamicsWorld . . . 919
10.185.3.9 GetPose . . . 919
10.185.3.10 GetType . . . 920
10.185.3.11 GetTypeString . . . 920
10.185.3.12 GetTypeString . . . 920
10.185.3.13 Init . . . 920
10.185.3.14 InitForThread . . . 920
10.185.3.15 InitModel . . . 921
10.185.3.16 Load . . . 921
10.185.3.17 OnPhysicsMsg . . . 921
10.185.3.18 OnRequest . . . 921
10.185.3.19 Pose2Transform . . . 921
10.185.3.20 QuadToQuad . . . 922
10.185.3.21 QuadToQuad . . . 922
10.185.3.22 Reset . . . 922
10.185.3.23 SetGravity . . . 922
10.185.3.24 SetSeed . . . 922
10.185.3.25 Transform2Pose . . . 923
10.185.3.26 UpdateCollision . . . 923

10.185.3.27 UpdatePhysics . . . 923
10.185.3.28 Vec3ToVector3 . . . 923
10.185.3.29 Vector3ToVec3 . . . 923
10.185.4 Member Data Documentation . . . 924
10.185.4.1 contact . . . 924
10.185.4.2 discreteForces . . . 924
10.185.4.3 forces . . . 924
10.185.4.4 gravity . . . 924
10.185.4.5 integ . . . 924
10.185.4.6 matter . . . 924
10.185.4.7 simbodyPhysicsInitialized . . . 924
10.185.4.8 simbodyPhysicsStepped . . . 924
10.185.4.9 system . . . 924
10.185.4.10 tracker . . . 924
10.186 gazebo::physics::SimbodyPlaneShape Class Reference . . . 924
10.186.1 Detailed Description . . . 925
10.186.2 Constructor & Destructor Documentation . . . 926
10.186.2.1 SimbodyPlaneShape . . . 926
10.186.2.2 ~SimbodyPlaneShape . . . 926
10.186.3 Member Function Documentation . . . 926
10.186.3.1 CreatePlane . . . 926
10.186.3.2 SetAltitude . . . 926
10.187 gazebo::physics::SimbodyRayShape Class Reference . . . 926
10.187.1 Detailed Description . . . 928
10.187.2 Constructor & Destructor Documentation . . . 928
10.187.2.1 SimbodyRayShape . . . 928
10.187.2.2 SimbodyRayShape . . . 928
10.187.2.3 ~SimbodyRayShape . . . 928
10.187.3 Member Function Documentation . . . 928
10.187.3.1 GetIntersection . . . 928
10.187.3.2 SetPoints . . . 928
10.187.3.3 Update . . . 929
10.188 gazebo::physics::SimbodyScrewJoint Class Reference . . . 929
10.188.1 Detailed Description . . . 931
10.188.2 Constructor & Destructor Documentation . . . 931
10.188.2.1 SimbodyScrewJoint . . . 931
10.188.2.2 ~SimbodyScrewJoint . . . 932

10.188.3 Member Function Documentation . . . 932
10.188.3.1 GetAngleImpl . . . 932
10.188.3.2 GetGlobalAxis . . . 932
10.188.3.3 GetHighStop . . . 932
10.188.3.4 GetLowStop . . . 933
10.188.3.5 GetMaxForce . . . 933
10.188.3.6 GetThreadPitch . . . 933
10.188.3.7 GetVelocity . . . 933
10.188.3.8 Init . . . 934
10.188.3.9 Load . . . 934
10.188.3.10 SetAxis . . . 934
10.188.3.11 SetDamping . . . 934
10.188.3.12 SetForceImpl . . . 935
10.188.3.13 SetHighStop . . . 935
10.188.3.14 SetLowStop . . . 935
10.188.3.15 SetMaxForce . . . 935
10.188.3.16 SetThreadPitch . . . 936
10.188.3.17 SetVelocity . . . 936
10.189 gazebo::physics::SimbodySliderJoint Class Reference . . . 936
10.189.1 Detailed Description . . . 938
10.189.2 Constructor & Destructor Documentation . . . 938
10.189.2.1 SimbodySliderJoint . . . 938
10.189.2.2 ~SimbodySliderJoint . . . 938
10.189.3 Member Function Documentation . . . 939
10.189.3.1 GetAngleImpl . . . 939
10.189.3.2 GetGlobalAxis . . . 939
10.189.3.3 GetHighStop . . . 939
10.189.3.4 GetLowStop . . . 939
10.189.3.5 GetMaxForce . . . 940
10.189.3.6 GetVelocity . . . 940
10.189.3.7 Load . . . 940
10.189.3.8 SetAxis . . . 941
10.189.3.9 SetDamping . . . 941
10.189.3.10 SetForceImpl . . . 941
10.189.3.11 SetHighStop . . . 941
10.189.3.12 SetLowStop . . . 942
10.189.3.13 SetMaxForce . . . 942

10.189.3.19SetVelocity . . . 942
10.190gazebo::physics::SimbodySphereShape Class Reference . . . 942
10.190.1Detailed Description . . . 943
10.190.2Constructor & Destructor Documentation . . . 943
10.190.2.1SimbodySphereShape . . . 944
10.190.2.2~SimbodySphereShape . . . 944
10.190.3Member Function Documentation . . . 944
10.190.3.1SetRadius . . . 944
10.191gazebo::physics::SimbodyUniversalJoint Class Reference . . . 944
10.191.1Detailed Description . . . 946
10.191.2Constructor & Destructor Documentation . . . 946
10.191.2.1SimbodyUniversalJoint . . . 946
10.191.2.2~SimbodyUniversalJoint . . . 947
10.191.3Member Function Documentation . . . 947
10.191.3.1GetAnchor . . . 947
10.191.3.2GetAngleImpl . . . 947
10.191.3.3GetAxis . . . 947
10.191.3.4GetGlobalAxis . . . 947
10.191.3.5GetHighStop . . . 948
10.191.3.6GetLowStop . . . 948
10.191.3.7GetMaxForce . . . 948
10.191.3.8GetVelocity . . . 948
10.191.3.9Init . . . 949
10.191.3.10Load . . . 949
10.191.3.11SetAxis . . . 949
10.191.3.12SetDamping . . . 949
10.191.3.13SetForceImpl . . . 950
10.191.3.14SetHighStop . . . 950
10.191.3.15SetLowStop . . . 950
10.191.3.16SetMaxForce . . . 950
10.191.3.17SetVelocity . . . 951
10.192SingletonT< T > Class Template Reference . . . 951
10.192.1Detailed Description . . . 953
10.192.2Constructor & Destructor Documentation . . . 953
10.192.2.1SingletonT . . . 953
10.192.2.2~SingletonT . . . 953
10.192.3Member Function Documentation . . . 953

10.192.3.1 Instance . . . 953
10.193 gazebo::common::Skeleton Class Reference . . . 953
10.193.1 Detailed Description . . . 955
10.193.2 Constructor & Destructor Documentation . . . 955
10.193.2.1 Skeleton . . . 955
10.193.2.2 Skeleton . . . 955
10.193.2.3 ~Skeleton . . . 955
10.193.3 Member Function Documentation . . . 955
10.193.3.1 AddAnimation . . . 955
10.193.3.2 AddVertNodeWeight . . . 956
10.193.3.3 BuildNodeMap . . . 956
10.193.3.4 GetAnimation . . . 956
10.193.3.5 GetBindShapeTransform . . . 956
10.193.3.6 GetNodeByHandle . . . 956
10.193.3.7 GetNodeById . . . 956
10.193.3.8 GetNodeByName . . . 957
10.193.3.9 GetNodes . . . 957
10.193.3.10 GetNumAnimations . . . 957
10.193.3.11 GetNumJoints . . . 957
10.193.3.12 GetNumNodes . . . 957
10.193.3.13 GetNumVertNodeWeights . . . 958
10.193.3.14 GetRootNode . . . 958
10.193.3.15 GetVertNodeWeight . . . 958
10.193.3.16 PrintTransforms . . . 958
10.193.3.17 Scale . . . 958
10.193.3.18 SetBindShapeTransform . . . 958
10.193.3.19 SetNumVertAttached . . . 959
10.193.3.20 SetRootNode . . . 959
10.193.4 Member Data Documentation . . . 959
10.193.4.1 anims . . . 959
10.193.4.2 bindShapeTransform . . . 959
10.193.4.3 nodes . . . 959
10.193.4.4 rawNW . . . 959
10.193.4.5 root . . . 959
10.194 gazebo::common::SkeletonAnimation Class Reference . . . 960
10.194.1 Detailed Description . . . 961
10.194.2 Constructor & Destructor Documentation . . . 961

10.194.2.1 SkeletonAnimation . . . 961
10.194.2.2 ~SkeletonAnimation . . . 961
10.194.3 Member Function Documentation . . . 961
10.194.3.1 AddKeyFrame . . . 961
10.194.3.2 AddKeyFrame . . . 961
10.194.3.3 GetLength . . . 961
10.194.3.4 GetName . . . 962
10.194.3.5 GetNodeCount . . . 962
10.194.3.6 GetNodePoseAt . . . 962
10.194.3.7 GetPoseAt . . . 962
10.194.3.8 GetPoseAtX . . . 963
10.194.3.9 HasNode . . . 963
10.194.3.10 Scale . . . 963
10.194.3.11 SetName . . . 963
10.194.4 Member Data Documentation . . . 963
10.194.4.1 animations . . . 963
10.194.4.2 length . . . 964
10.194.4.3 name . . . 964
10.195 gazebo::common::SkeletonNode Class Reference . . . 964
10.195.1 Detailed Description . . . 966
10.195.2 Member Enumeration Documentation . . . 966
10.195.2.1 SkeletonNodeType . . . 966
10.195.3 Constructor & Destructor Documentation . . . 966
10.195.3.1 SkeletonNode . . . 966
10.195.3.2 SkeletonNode . . . 967
10.195.3.3 ~SkeletonNode . . . 967
10.195.4 Member Function Documentation . . . 967
10.195.4.1 AddChild . . . 967
10.195.4.2 AddRawTransform . . . 967
10.195.4.3 GetChild . . . 967
10.195.4.4 GetChildById . . . 967
10.195.4.5 GetChildByName . . . 968
10.195.4.6 GetChildCount . . . 968
10.195.4.7 GetHandle . . . 968
10.195.4.8 GetId . . . 968
10.195.4.9 GetInverseBindTransform . . . 968
10.195.4.10 GetModelTransform . . . 969

10.195.4.11 GetName . . . 969
10.195.4.12 GetNumRawTrans . . . 969
10.195.4.13 GetParent . . . 969
10.195.4.14 GetRawTransform . . . 969
10.195.4.15 GetRawTransforms . . . 969
10.195.4.16 GetTransform . . . 970
10.195.4.17 GetTransforms . . . 970
10.195.4.18 IsJoint . . . 970
10.195.4.19 IsRootNode . . . 970
10.195.4.20 Reset . . . 970
10.195.4.21 SetHandle . . . 970
10.195.4.22 SetId . . . 971
10.195.4.23 SetInitialTransform . . . 971
10.195.4.24 SetInverseBindTransform . . . 971
10.195.4.25 SetModelTransform . . . 971
10.195.4.26 SetName . . . 971
10.195.4.27 SetParent . . . 971
10.195.4.28 SetTransform . . . 972
10.195.4.29 SetType . . . 972
10.195.4.30 UpdateChildrenTransforms . . . 972
10.195.5 Member Data Documentation . . . 972
10.195.5.1 children . . . 972
10.195.5.2 handle . . . 972
10.195.5.3 id . . . 972
10.195.5.4 initialTransform . . . 972
10.195.5.5 invBindTransform . . . 972
10.195.5.6 modelTransform . . . 972
10.195.5.7 name . . . 973
10.195.5.8 parent . . . 973
10.195.5.9 rawTransforms . . . 973
10.195.5.10 transform . . . 973
10.195.5.11 type . . . 973
10.196 gazebo::physics::SliderJoint< T > Class Template Reference . . . 973
10.196.1 Detailed Description . . . 974
10.196.2 Constructor & Destructor Documentation . . . 974
10.196.2.1 SliderJoint . . . 974
10.196.2.2 ~SliderJoint . . . 974

10.196.3 Member Function Documentation . . . 974
10.196.3.1 GetAnchor . . . 974
10.196.3.2 GetAngleCount . . . 975
10.196.3.3 Load . . . 975
10.196.3.4 SetAnchor . . . 975
10.196.4 Member Data Documentation . . . 975
10.196.4.1 fakeAnchor . . . 975
10.197 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference . . . 975
10.197.1 Detailed Description . . . 976
10.197.2 Constructor & Destructor Documentation . . . 977
10.197.2.1 SM2Profile . . . 977
10.197.2.2 ~SM2Profile . . . 977
10.197.3 Member Function Documentation . . . 977
10.197.3.1 addTechnique . . . 977
10.197.3.2 generate . . . 977
10.197.3.3 generateForCompositeMap . . . 977
10.197.3.4 UpdateParams . . . 977
10.197.3.5 UpdateParamsForCompositeMap . . . 977
10.198 gazebo::sensors::SonarSensor Class Reference . . . 977
10.198.1 Detailed Description . . . 979
10.198.2 Constructor & Destructor Documentation . . . 979
10.198.2.1 SonarSensor . . . 979
10.198.2.2 ~SonarSensor . . . 979
10.198.3 Member Function Documentation . . . 979
10.198.3.1 ConnectUpdate . . . 979
10.198.3.2 DisconnectUpdate . . . 979
10.198.3.3 Fini . . . 980
10.198.3.4 GetRadius . . . 980
10.198.3.5 GetRange . . . 980
10.198.3.6 GetRangeMax . . . 980
10.198.3.7 GetRangeMin . . . 980
10.198.3.8 GetTopic . . . 981
10.198.3.9 Init . . . 981
10.198.3.10 IsActive . . . 981
10.198.3.11 Load . . . 981
10.198.3.12 UpdateImpl . . . 981
10.198.4 Member Data Documentation . . . 982

10.198.4.1update . . . 982
10.199gazebo::rendering::SonarVisual Class Reference . . . 982
10.199.1Detailed Description . . . 983
10.199.2Constructor & Destructor Documentation . . . 983
10.199.2.1SonarVisual . . . 983
10.199.2.2~SonarVisual . . . 983
10.199.3Member Function Documentation . . . 983
10.199.3.1Load . . . 983
10.200Joint_TEST::SpawnJointOptions Class Reference . . . 983
10.200.1Detailed Description . . . 984
10.200.2Constructor & Destructor Documentation . . . 984
10.200.2.1SpawnJointOptions . . . 984
10.200.2.2~SpawnJointOptions . . . 984
10.200.3Member Data Documentation . . . 984
10.200.3.1axis . . . 984
10.200.3.2childLinkPose . . . 985
10.200.3.3jointPose . . . 985
10.200.3.4modelPose . . . 985
10.200.3.5noLinkPose . . . 985
10.200.3.6parentLinkPose . . . 985
10.200.3.7type . . . 985
10.200.3.8wait . . . 985
10.200.3.9worldChild . . . 985
10.200.3.10worldParent . . . 985
10.201gazebo::physics::SphereShape Class Reference . . . 986
10.201.1Detailed Description . . . 987
10.201.2Constructor & Destructor Documentation . . . 987
10.201.2.1SphereShape . . . 987
10.201.2.2~SphereShape . . . 987
10.201.3Member Function Documentation . . . 987
10.201.3.1FillMsg . . . 987
10.201.3.2GetRadius . . . 987
10.201.3.3Init . . . 988
10.201.3.4ProcessMsg . . . 988
10.201.3.5SetRadius . . . 988
10.201.3.6SetScale . . . 988
10.202gazebo::common::SphericalCoordinates Class Reference . . . 988

10.202.1 Detailed Description . . . 990
10.202.2 Member Enumeration Documentation . . . 990
10.202.2.1 SurfaceType . . . 990
10.202.3 Constructor & Destructor Documentation . . . 990
10.202.3.1 SphericalCoordinates . . . 990
10.202.3.2 SphericalCoordinates . . . 990
10.202.3.3 SphericalCoordinates . . . 990
10.202.3.4 ~SphericalCoordinates . . . 991
10.202.4 Member Function Documentation . . . 991
10.202.4.1 Convert . . . 991
10.202.4.2 GetElevationReference . . . 991
10.202.4.3 GetHeadingOffset . . . 991
10.202.4.4 GetLatitudeReference . . . 991
10.202.4.5 GetLongitudeReference . . . 991
10.202.4.6 GetSurfaceType . . . 992
10.202.4.7 GlobalFromLocal . . . 992
10.202.4.8 SetElevationReference . . . 992
10.202.4.9 SetHeadingOffset . . . 992
10.202.4.10 SetLatitudeReference . . . 992
10.202.4.11 SetLongitudeReference . . . 992
10.202.4.12 SetSurfaceType . . . 993
10.202.4.13 SphericalFromLocal . . . 993
10.203 gazebo::math::Spline Class Reference . . . 993
10.203.1 Detailed Description . . . 994
10.203.2 Constructor & Destructor Documentation . . . 994
10.203.2.1 Spline . . . 994
10.203.2.2 ~Spline . . . 994
10.203.3 Member Function Documentation . . . 995
10.203.3.1 AddPoint . . . 995
10.203.3.2 Clear . . . 995
10.203.3.3 GetPoint . . . 995
10.203.3.4 GetPointCount . . . 995
10.203.3.5 GetTangent . . . 995
10.203.3.6 GetTension . . . 995
10.203.3.7 Interpolate . . . 996
10.203.3.8 Interpolate . . . 996
10.203.3.9 RecalcTangents . . . 996

10.203.3.13SetAutoCalculate . . . 996
10.203.3.14SetTension . . . 997
10.203.3.15UpdatePoint . . . 997
10.203.4Member Data Documentation . . . 997
10.203.4.1autoCalc . . . 997
10.203.4.2coeffs . . . 997
10.203.4.3points . . . 997
10.203.4.4tangents . . . 997
10.203.4.5tension . . . 997
10.204gazebo::physics::State Class Reference . . . 998
10.204.1Detailed Description . . . 999
10.204.2Constructor & Destructor Documentation . . . 999
10.204.2.1State . . . 999
10.204.2.2State . . . 999
10.204.2.3~State . . . 999
10.204.3Member Function Documentation . . . 1000
10.204.3.1GetName . . . 1000
10.204.3.2GetRealTime . . . 1000
10.204.3.3GetSimTime . . . 1000
10.204.3.4GetWallTime . . . 1000
10.204.3.5Load . . . 1000
10.204.3.6operator- . . . 1001
10.204.3.7operator= . . . 1001
10.204.3.8SetName . . . 1001
10.204.3.9SetRealTime . . . 1001
10.204.3.10SetSimTime . . . 1001
10.204.3.11SetWallTime . . . 1002
10.204.4Member Data Documentation . . . 1002
10.204.4.1name . . . 1002
10.204.4.2realTime . . . 1002
10.204.4.3simTime . . . 1002
10.204.4.4wallTime . . . 1002
10.205gazebo::common::STLLoader Class Reference . . . 1002
10.205.1Detailed Description . . . 1003
10.205.2Constructor & Destructor Documentation . . . 1003
10.205.2.1STLLoader . . . 1003
10.205.2.2~STLLoader . . . 1003

10.205.3 Member Function Documentation . . . 1003
10.205.3.1 Load . . . 1003
10.206 gazebo::common::SubMesh Class Reference . . . 1004
10.206.1 Detailed Description . . . 1006
10.206.2 Member Enumeration Documentation . . . 1006
10.206.2.1 PrimitiveType . . . 1006
10.206.3 Constructor & Destructor Documentation . . . 1006
10.206.3.1 SubMesh . . . 1006
10.206.3.2 SubMesh . . . 1007
10.206.3.3 ~SubMesh . . . 1007
10.206.4 Member Function Documentation . . . 1007
10.206.4.1 AddIndex . . . 1007
10.206.4.2 AddNodeAssignment . . . 1007
10.206.4.3 AddNormal . . . 1007
10.206.4.4 AddNormal . . . 1007
10.206.4.5 AddTexCoord . . . 1008
10.206.4.6 AddVertex . . . 1008
10.206.4.7 AddVertex . . . 1008
10.206.4.8 Center . . . 1008
10.206.4.9 CopyNormals . . . 1008
10.206.4.10 CopyVertices . . . 1008
10.206.4.11 FillArrays . . . 1009
10.206.4.12 GenSphericalTexCoord . . . 1009
10.206.4.13 GetIndex . . . 1009
10.206.4.14 GetIndexCount . . . 1009
10.206.4.15 GetMaterialIndex . . . 1009
10.206.4.16 GetMax . . . 1009
10.206.4.17 GetMaxIndex . . . 1009
10.206.4.18 GetMin . . . 1010
10.206.4.19 GetName . . . 1010
10.206.4.20 GetNodeAssignment . . . 1010
10.206.4.21 GetNodeAssignmentsCount . . . 1010
10.206.4.22 GetNormal . . . 1010
10.206.4.23 GetNormalCount . . . 1010
10.206.4.24 GetPrimitiveType . . . 1010
10.206.4.25 GetTexCoord . . . 1011
10.206.4.26 GetTexCoordCount . . . 1011

10.206.4.27 GetVertex . . . 1011
10.206.4.28 GetVertexCount . . . 1011
10.206.4.29 GetVertexIndex . . . 1011
10.206.4.30 HasVertex . . . 1011
10.206.4.31 RecalculateNormals . . . 1012
10.206.4.32 Scale . . . 1012
10.206.4.33 SetIndexCount . . . 1012
10.206.4.34 SetMaterialIndex . . . 1012
10.206.4.35 SetName . . . 1012
10.206.4.36 SetNormal . . . 1012
10.206.4.37 SetNormalCount . . . 1013
10.206.4.38 SetPrimitiveType . . . 1013
10.206.4.39 SetScale . . . 1013
10.206.4.40 SetSubMeshCenter . . . 1013
10.206.4.41 SetTexCoord . . . 1013
10.206.4.42 SetTexCoordCount . . . 1014
10.206.4.43 SetVertex . . . 1014
10.206.4.44 SetVertexCount . . . 1014
10.206.4.45 Translate . . . 1014
10.207 gazebo::transport::SubscribeOptions Class Reference . . . 1014
10.207.1 Detailed Description . . . 1015
10.207.2 Constructor & Destructor Documentation . . . 1015
10.207.2.1 SubscribeOptions . . . 1015
10.207.3 Member Function Documentation . . . 1015
10.207.3.1 GetLatching . . . 1015
10.207.3.2 GetMsgType . . . 1015
10.207.3.3 GetNode . . . 1015
10.207.3.4 GetTopic . . . 1016
10.207.3.5 Init . . . 1016
10.207.3.6 Init . . . 1016
10.208 gazebo::transport::Subscriber Class Reference . . . 1016
10.208.1 Detailed Description . . . 1017
10.208.2 Constructor & Destructor Documentation . . . 1017
10.208.2.1 Subscriber . . . 1017
10.208.2.2 ~Subscriber . . . 1017
10.208.3 Member Function Documentation . . . 1017
10.208.3.1 GetCallbackId . . . 1017

10.208.3.2 GetTopic . . . 1017
10.208.3.3 SetCallbackId . . . 1017
10.208.3.4 Unsubscribe . . . 1017
10.209 gazebo::transport::SubscriptionTransport Class Reference . . . 1018
10.209.1 Detailed Description . . . 1018
10.209.2 Constructor & Destructor Documentation . . . 1019
10.209.2.1 SubscriptionTransport . . . 1019
10.209.2.2 ~SubscriptionTransport . . . 1019
10.209.3 Member Function Documentation . . . 1019
10.209.3.1 GetConnection . . . 1019
10.209.3.2 HandleData . . . 1019
10.209.3.3 HandleMessage . . . 1019
10.209.3.4 Init . . . 1020
10.209.3.5 IsLocal . . . 1020
10.210 gazebo::physics::SurfaceParams Class Reference . . . 1020
10.210.1 Detailed Description . . . 1021
10.210.2 Constructor & Destructor Documentation . . . 1021
10.210.2.1 SurfaceParams . . . 1021
10.210.2.2 ~SurfaceParams . . . 1022
10.210.3 Member Function Documentation . . . 1022
10.210.3.1 FillMsg . . . 1022
10.210.3.2 Load . . . 1022
10.210.3.3 ProcessMsg . . . 1022
10.210.4 Member Data Documentation . . . 1022
10.210.4.1 bounce . . . 1022
10.210.4.2 bounceThreshold . . . 1022
10.210.4.3 cfm . . . 1022
10.210.4.4 collideWithoutContact . . . 1023
10.210.4.5 collideWithoutContactBitmask . . . 1023
10.210.4.6 erp . . . 1023
10.210.4.7 fdir1 . . . 1023
10.210.4.8 kd . . . 1023
10.210.4.9 kp . . . 1023
10.210.4.10 maxVel . . . 1024
10.210.4.11 minDepth . . . 1024
10.210.4.12 mu1 . . . 1024
10.210.4.13 mu2 . . . 1024

10.210.4.14 slip1 . . . 1024
10.210.4.15 slip2 . . . 1025
10.211 gazebo::common::SystemPaths Class Reference . . . 1025
10.211.1 Detailed Description . . . 1026
10.211.2 Member Function Documentation . . . 1027
10.211.2.1 AddGazeboPaths . . . 1027
10.211.2.2 AddModelPaths . . . 1027
10.211.2.3 AddOgrePaths . . . 1027
10.211.2.4 AddPluginPaths . . . 1027
10.211.2.5 AddSearchPathSuffix . . . 1027
10.211.2.6 ClearGazeboPaths . . . 1027
10.211.2.7 ClearModelPaths . . . 1028
10.211.2.8 ClearOgrePaths . . . 1028
10.211.2.9 ClearPluginPaths . . . 1028
10.211.2.10 FindFile . . . 1028
10.211.2.11 FindFileURI . . . 1028
10.211.2.12 GetGazeboPaths . . . 1028
10.211.2.13 GetLogPath . . . 1028
10.211.2.14 GetModelPaths . . . 1029
10.211.2.15 GetOgrePaths . . . 1029
10.211.2.16 GetPluginPaths . . . 1029
10.211.2.17 GetWorldPathExtension . . . 1029
10.211.3 Member Data Documentation . . . 1029
10.211.3.1 gazeboPathsFromEnv . . . 1029
10.211.3.2 modelPathsFromEnv . . . 1029
10.211.3.3 ogrePathsFromEnv . . . 1029
10.211.3.4 pluginPathsFromEnv . . . 1030
10.212 gazebo::SystemPlugin Class Reference . . . 1030
10.212.1 Detailed Description . . . 1030
10.212.2 Constructor & Destructor Documentation . . . 1031
10.212.2.1 SystemPlugin . . . 1031
10.212.2.2 ~SystemPlugin . . . 1031
10.212.3 Member Function Documentation . . . 1031
10.212.3.1 Init . . . 1031
10.212.3.2 Load . . . 1031
10.212.3.3 Reset . . . 1031
10.213 gazebo::common::Time Class Reference . . . 1031

10.213.1 Detailed Description . . . 1035
10.213.2 Constructor & Destructor Documentation . . . 1035
10.213.2.1 Time . . . 1035
10.213.2.2 Time . . . 1035
10.213.2.3 Time . . . 1035
10.213.2.4 Time . . . 1036
10.213.2.5 Time . . . 1036
10.213.2.6 Time . . . 1036
10.213.2.7 ∼Time . . . 1036
10.213.3 Member Function Documentation . . . 1036
10.213.3.1 Double . . . 1036
10.213.3.2 Float . . . 1036
10.213.3.3 GetWallTime . . . 1037
10.213.3.4 GetWallTimeAsISOString . . . 1037
10.213.3.5 MicToNano . . . 1037
10.213.3.6 MilToNano . . . 1037
10.213.3.7 MSleep . . . 1037
10.213.3.8 NSleep . . . 1038
10.213.3.9 operator!= . . . 1038
10.213.3.10 operator!= . . . 1038
10.213.3.11 operator!= . . . 1038
10.213.3.12 operator!= . . . 1039
10.213.3.13 operator∗ . . . 1039
10.213.3.14 operator∗ . . . 1039
10.213.3.15 operator∗ . . . 1039
10.213.3.16 operator∗= . . . 1040
10.213.3.17 operator∗= . . . 1040
10.213.3.18 operator∗= . . . 1040
10.213.3.19 operator+ . . . 1040
10.213.3.20 operator+ . . . 1041
10.213.3.21 operator+ . . . 1041
10.213.3.22 operator+= . . . 1041
10.213.3.23 operator+= . . . 1041
10.213.3.24 operator+= . . . 1042
10.213.3.25 operator- . . . 1042
10.213.3.26 operator- . . . 1042
10.213.3.27 operator- . . . 1042

10.213.3.28operator-= . . . 1043
10.213.3.29operator-= . . . 1043
10.213.3.30operator-= . . . 1043
10.213.3.31operator/ . . . 1043
10.213.3.32operator/ . . . 1044
10.213.3.33operator/ . . . 1044
10.213.3.34operator/= . . . 1044
10.213.3.35operator/= . . . 1044
10.213.3.36operator/= . . . 1045
10.213.3.37operator< . . . 1045
10.213.3.38operator< . . . 1045
10.213.3.39operator< . . . 1045
10.213.3.40operator< . . . 1046
10.213.3.41operator<= . . . 1046
10.213.3.42operator<= . . . 1046
10.213.3.43operator<= . . . 1046
10.213.3.44operator<= . . . 1047
10.213.3.45operator= . . . 1047
10.213.3.46operator= . . . 1047
10.213.3.47operator= . . . 1047
10.213.3.48operator== . . . 1048
10.213.3.49operator== . . . 1048
10.213.3.50operator== . . . 1048
10.213.3.51operator== . . . 1048
10.213.3.52operator> . . . 1049
10.213.3.53operator> . . . 1049
10.213.3.54operator> . . . 1049
10.213.3.55operator> . . . 1049
10.213.3.56operator>= . . . 1050
10.213.3.57operator>= . . . 1050
10.213.3.58operator>= . . . 1050
10.213.3.59operator>= . . . 1050
10.213.3.60SecToNano . . . 1051
10.213.3.61Set . . . 1051
10.213.3.62Set . . . 1051
10.213.3.63SetToWallTime . . . 1051
10.213.3.64Sleep . . . 1051

10.213.4 Friends And Related Function Documentation . . . 1052
10.213.4.1 operator<< . . . 1052
10.213.4.2 operator>> . . . 1052
10.213.5 Member Data Documentation . . . 1052
10.213.5.1 nsec . . . 1052
10.213.5.2 sec . . . 1052
10.213.5.3 Zero . . . 1052
10.214 gazebo::common::Timer Class Reference . . . 1053
10.214.1 Detailed Description . . . 1053
10.214.2 Constructor & Destructor Documentation . . . 1054
10.214.2.1 Timer . . . 1054
10.214.2.2 ~Timer . . . 1054
10.214.3 Member Function Documentation . . . 1054
10.214.3.1 GetElapsed . . . 1054
10.214.3.2 GetRunning . . . 1054
10.214.3.3 Start . . . 1054
10.214.3.4 Stop . . . 1054
10.214.4 Friends And Related Function Documentation . . . 1054
10.214.4.1 operator<< . . . 1054
10.215 gazebo::transport::TopicManager Class Reference . . . 1055
10.215.1 Detailed Description . . . 1056
10.215.2 Member Typedef Documentation . . . 1056
10.215.2.1 SubNodeMap . . . 1056
10.215.3 Member Function Documentation . . . 1057
10.215.3.1 AddNode . . . 1057
10.215.3.2 AddNodeToProcess . . . 1057
10.215.3.3 Advertise . . . 1057
10.215.3.4 ClearBuffers . . . 1057
10.215.3.5 ConnectPubToSub . . . 1057
10.215.3.6 ConnectSubscribers . . . 1058
10.215.3.7 ConnectSubToPub . . . 1058
10.215.3.8 DisconnectPubFromSub . . . 1058
10.215.3.9 DisconnectSubFromPub . . . 1058
10.215.3.10 FindPublication . . . 1058
10.215.3.11 Fini . . . 1059
10.215.3.12 GetTopicNamespaces . . . 1059
10.215.3.13 Init . . . 1059

10.215.3.14 IsAdvertised . . . 1059
10.215.3.15 PauseIncoming . . . 1059
10.215.3.16 ProcessNodes . . . 1059
10.215.3.17 Publish . . . 1060
10.215.3.18 RegisterTopicNamespace . . . 1060
10.215.3.19 RemoveNode . . . 1060
10.215.3.20 Subscribe . . . 1060
10.215.3.21 Unadvertise . . . 1060
10.215.3.22 Unsubscribe . . . 1061
10.215.3.23 UpdatePublications . . . 1061
10.216 gazebo::physics::TrajectoryInfo Struct Reference . . . 1061
10.216.1 Member Data Documentation . . . 1062
10.216.1.1 duration . . . 1062
10.216.1.2 endTime . . . 1062
10.216.1.3 id . . . 1062
10.216.1.4 startTime . . . 1062
10.216.1.5 translated . . . 1062
10.216.1.6 type . . . 1062
10.217 gazebo::rendering::TransmitterVisual Class Reference . . . 1062
10.217.1 Detailed Description . . . 1063
10.217.2 Constructor & Destructor Documentation . . . 1063
10.217.2.1 TransmitterVisual . . . 1063
10.217.2.2 ~TransmitterVisual . . . 1063
10.217.3 Member Function Documentation . . . 1063
10.217.3.1 Load . . . 1063
10.217.3.2 Update . . . 1063
10.218 gazebo::physics::UniversalJoint< T > Class Template Reference . . . 1064
10.218.1 Detailed Description . . . 1064
10.218.2 Constructor & Destructor Documentation . . . 1064
10.218.2.1 UniversalJoint . . . 1064
10.218.2.2 ~UniversalJoint . . . 1065
10.218.3 Member Function Documentation . . . 1065
10.218.3.1 GetAngleCount . . . 1065
10.218.3.2 Load . . . 1065
10.219 gazebo::common::UpdateInfo Class Reference . . . 1065
10.219.1 Detailed Description . . . 1065
10.219.2 Member Data Documentation . . . 1065

10.219.2.1realTime . . . 1065
10.219.2.2simTime . . . 1066
10.219.2.3worldName . . . 1066
10.220gazebo::rendering::UserCamera Class Reference . . . 1066
10.220.1Detailed Description . . . 1068
10.220.2Constructor & Destructor Documentation . . . 1068
10.220.2.1UserCamera . . . 1068
10.220.2.2~UserCamera . . . 1068
10.220.3Member Function Documentation . . . 1068
10.220.3.1AnimationComplete . . . 1068
10.220.3.2AttachToVisualImpl . . . 1069
10.220.3.3EnableViewController . . . 1069
10.220.3.4Fini . . . 1069
10.220.3.5GetAvgFPS . . . 1069
10.220.3.6GetGUIOverlay . . . 1069
10.220.3.7GetImageHeight . . . 1070
10.220.3.8GetImageWidth . . . 1070
10.220.3.9GetTriangleCount . . . 1070
10.220.3.10GetViewControllerTypeString . . . 1070
10.220.3.11GetVisual . . . 1070
10.220.3.12GetVisual . . . 1071
10.220.3.13HandleKeyPressEvent . . . 1071
10.220.3.14HandleKeyReleaseEvent . . . 1071
10.220.3.15HandleMouseEvent . . . 1071
10.220.3.16Init . . . 1071
10.220.3.17Load . . . 1071
10.220.3.18Load . . . 1072
10.220.3.19MoveToPosition . . . 1072
10.220.3.20MoveToVisual . . . 1072
10.220.3.21MoveToVisual . . . 1072
10.220.3.22PostRender . . . 1072
10.220.3.23Resize . . . 1073
10.220.3.24SetFocalPoint . . . 1073
10.220.3.25SetRenderTarget . . . 1073
10.220.3.26SetViewController . . . 1073
10.220.3.27SetViewController . . . 1073
10.220.3.28SetViewportDimensions . . . 1073

10.220.3.29 SetWorldPose . . . 1074
10.220.3.30 TrackVisualImpl . . . 1074
10.220.3.31 Update . . . 1074
10.221 gazebo::math::Vector2d Class Reference . . . 1074
10.221.1 Detailed Description . . . 1076
10.221.2 Constructor & Destructor Documentation . . . 1076
10.221.2.1 Vector2d . . . 1076
10.221.2.2 Vector2d . . . 1076
10.221.2.3 Vector2d . . . 1076
10.221.2.4 ~Vector2d . . . 1077
10.221.3 Member Function Documentation . . . 1077
10.221.3.1 Cross . . . 1077
10.221.3.2 Distance . . . 1077
10.221.3.3 IsFinite . . . 1077
10.221.3.4 Normalize . . . 1077
10.221.3.5 operator!= . . . 1077
10.221.3.6 operator* . . . 1078
10.221.3.7 operator* . . . 1078
10.221.3.8 operator*= . . . 1078
10.221.3.9 operator*= . . . 1078
10.221.3.10 operator+ . . . 1079
10.221.3.11 operator+= . . . 1079
10.221.3.12 operator- . . . 1079
10.221.3.13 operator-= . . . 1079
10.221.3.14 operator/ . . . 1080
10.221.3.15 operator/ . . . 1080
10.221.3.16 operator/= . . . 1080
10.221.3.17 operator/= . . . 1080
10.221.3.18 operator= . . . 1081
10.221.3.19 operator= . . . 1081
10.221.3.20 operator== . . . 1081
10.221.3.21 operator[] . . . 1081
10.221.3.22 Set . . . 1082
10.221.4 Friends And Related Function Documentation . . . 1082
10.221.4.1 operator<< . . . 1082
10.221.4.2 operator>> . . . 1082
10.221.5 Member Data Documentation . . . 1082

10.221.5.1x . . . 1082
10.221.5.2y . . . 1083
10.222gazebo::math::Vector2i Class Reference . . . 1083
10.222.1Detailed Description . . . 1084
10.222.2Constructor & Destructor Documentation . . . 1084
10.222.2.1Vector2i . . . 1084
10.222.2.2Vector2i . . . 1085
10.222.2.3Vector2i . . . 1085
10.222.2.4∼Vector2i . . . 1085
10.222.3Member Function Documentation . . . 1085
10.222.3.1Cross . . . 1085
10.222.3.2Distance . . . 1085
10.222.3.3IsFinite . . . 1086
10.222.3.4Normalize . . . 1086
10.222.3.5operator!= . . . 1086
10.222.3.6operator∗ . . . 1086
10.222.3.7operator∗ . . . 1086
10.222.3.8operator∗= . . . 1087
10.222.3.9operator∗= . . . 1087
10.222.3.10operator+ . . . 1087
10.222.3.11operator+= . . . 1087
10.222.3.12operator- . . . 1088
10.222.3.13operator-= . . . 1088
10.222.3.14operator/ . . . 1088
10.222.3.15operator/ . . . 1089
10.222.3.16operator/= . . . 1089
10.222.3.17operator/= . . . 1089
10.222.3.18operator= . . . 1090
10.222.3.19operator= . . . 1090
10.222.3.20operator== . . . 1090
10.222.3.21operator[] . . . 1090
10.222.3.22Set . . . 1090
10.222.4Friends And Related Function Documentation . . . 1091
10.222.4.1operator<< . . . 1091
10.222.4.2operator>> . . . 1091
10.222.5Member Data Documentation . . . 1091
10.222.5.1x . . . 1091

10.222.5.2 y . . . 1091
10.223 gazebo::math::Vector3 Class Reference . . . 1091
10.223.1 Detailed Description . . . 1094
10.223.2 Constructor & Destructor Documentation . . . 1094
10.223.2.1 Vector3 . . . 1094
10.223.2.2 Vector3 . . . 1095
10.223.2.3 Vector3 . . . 1095
10.223.2.4 ~Vector3 . . . 1095
10.223.3 Member Function Documentation . . . 1095
10.223.3.1 Correct . . . 1095
10.223.3.2 Cross . . . 1095
10.223.3.3 Distance . . . 1095
10.223.3.4 Distance . . . 1096
10.223.3.5 Dot . . . 1096
10.223.3.6 Equal . . . 1096
10.223.3.7 GetAbs . . . 1096
10.223.3.8 GetDistToLine . . . 1097
10.223.3.9 GetLength . . . 1097
10.223.3.10 GetMax . . . 1097
10.223.3.11 GetMin . . . 1097
10.223.3.12 GetNormal . . . 1097
10.223.3.13 GetPerpendicular . . . 1098
10.223.3.14 GetRounded . . . 1098
10.223.3.15 GetSquaredLength . . . 1098
10.223.3.16 GetSum . . . 1098
10.223.3.17 IsFinite . . . 1098
10.223.3.18 Normalize . . . 1098
10.223.3.19 operator!= . . . 1098
10.223.3.20 operator* . . . 1099
10.223.3.21 operator* . . . 1099
10.223.3.22 operator*= . . . 1099
10.223.3.23 operator*= . . . 1100
10.223.3.24 operator+ . . . 1100
10.223.3.25 operator+= . . . 1100
10.223.3.26 operator- . . . 1100
10.223.3.27 operator- . . . 1100
10.223.3.28 operator-= . . . 1101

10.223.3.29 operator/ . . . 1101
10.223.3.30 operator/ . . . 1101
10.223.3.31 operator/= . . . 1101
10.223.3.32 operator/= . . . 1102
10.223.3.33 operator= . . . 1102
10.223.3.34 operator= . . . 1102
10.223.3.35 operator== . . . 1102
10.223.3.36 operator[] . . . 1103
10.223.3.37 Round . . . 1103
10.223.3.38 Round . . . 1103
10.223.3.39 Set . . . 1103
10.223.3.40 SetToMax . . . 1103
10.223.3.41 SetToMin . . . 1103
10.223.4 Friends And Related Function Documentation . . . 1104
10.223.4.1 operator∗ . . . 1104
10.223.4.2 operator<< . . . 1104
10.223.4.3 operator>> . . . 1104
10.223.5 Member Data Documentation . . . 1104
10.223.5.1 One . . . 1104
10.223.5.2 UnitX . . . 1104
10.223.5.3 UnitY . . . 1105
10.223.5.4 UnitZ . . . 1105
10.223.5.5 x . . . 1105
10.223.5.6 y . . . 1105
10.223.5.7 z . . . 1105
10.223.5.8 Zero . . . 1105
10.224 gazebo::math::Vector4 Class Reference . . . 1105
10.224.1 Detailed Description . . . 1107
10.224.2 Constructor & Destructor Documentation . . . 1107
10.224.2.1 Vector4 . . . 1107
10.224.2.2 Vector4 . . . 1107
10.224.2.3 Vector4 . . . 1108
10.224.2.4 ∼Vector4 . . . 1108
10.224.3 Member Function Documentation . . . 1108
10.224.3.1 Distance . . . 1108
10.224.3.2 GetLength . . . 1108
10.224.3.3 GetSquaredLength . . . 1108

10.224.3.4 IsFinite . . . 1108
10.224.3.5 Normalize . . . 1108
10.224.3.6 operator!= . . . 1109
10.224.3.7 operator* . . . 1109
10.224.3.8 operator* . . . 1109
10.224.3.9 operator* . . . 1109
10.224.3.10 operator*= . . . 1110
10.224.3.11 operator*= . . . 1110
10.224.3.12 operator+ . . . 1110
10.224.3.13 operator+= . . . 1110
10.224.3.14 operator- . . . 1111
10.224.3.15 operator-= . . . 1111
10.224.3.16 operator/ . . . 1111
10.224.3.17 operator/ . . . 1112
10.224.3.18 operator/= . . . 1112
10.224.3.19 operator/= . . . 1112
10.224.3.20 operator= . . . 1112
10.224.3.21 operator= . . . 1113
10.224.3.22 operator== . . . 1113
10.224.3.23 operator[] . . . 1113
10.224.3.24 Set . . . 1113
10.224.4 Friends And Related Function Documentation . . . 1113
10.224.4.1 operator<< . . . 1114
10.224.4.2 operator>> . . . 1114
10.224.5 Member Data Documentation . . . 1114
10.224.5.1 w . . . 1114
10.224.5.2 x . . . 1114
10.224.5.3 y . . . 1114
10.224.5.4 z . . . 1114
10.225 gazebo::common::Video Class Reference . . . 1115
10.225.1 Detailed Description . . . 1115
10.225.2 Constructor & Destructor Documentation . . . 1115
10.225.2.1 Video . . . 1115
10.225.2.2 ~Video . . . 1115
10.225.3 Member Function Documentation . . . 1115
10.225.3.1 GetHeight . . . 1115
10.225.3.2 GetNextFrame . . . 1116

10.225.3.3 GetWidth . . . 1116
10.225.3.4 Load . . . 1116
10.226 gazebo::rendering::VideoVisual Class Reference . . . 1116
10.226.1 Detailed Description . . . 1117
10.226.2 Constructor & Destructor Documentation . . . 1117
10.226.2.1 VideoVisual . . . 1117
10.226.2.2 ~VideoVisual . . . 1118
10.227 gazebo::rendering::ViewController Class Reference . . . 1118
10.227.1 Detailed Description . . . 1119
10.227.2 Constructor & Destructor Documentation . . . 1119
10.227.2.1 ViewController . . . 1119
10.227.2.2 ~ViewController . . . 1119
10.227.3 Member Function Documentation . . . 1119
10.227.3.1 GetTypeString . . . 1119
10.227.3.2 HandleKeyPressEvent . . . 1119
10.227.3.3 HandleKeyReleaseEvent . . . 1120
10.227.3.4 HandleMouseEvent . . . 1120
10.227.3.5 Init . . . 1120
10.227.3.6 Init . . . 1120
10.227.3.7 SetEnabled . . . 1120
10.227.3.8 Update . . . 1121
10.227.4 Member Data Documentation . . . 1121
10.227.4.1 camera . . . 1121
10.227.4.2 enabled . . . 1121
10.227.4.3 typeString . . . 1121
10.228 gazebo::rendering::Visual Class Reference . . . 1121
10.228.1 Detailed Description . . . 1127
10.228.2 Constructor & Destructor Documentation . . . 1127
10.228.2.1 Visual . . . 1127
10.228.2.2 Visual . . . 1127
10.228.2.3 ~Visual . . . 1127
10.228.3 Member Function Documentation . . . 1127
10.228.3.1 AttachAxes . . . 1127
10.228.3.2 AttachLineVertex . . . 1127
10.228.3.3 AttachMesh . . . 1128
10.228.3.4 AttachObject . . . 1128
10.228.3.5 AttachVisual . . . 1128

10.228.3.6 ClearParent . . . 1128
10.228.3.7 Clone . . . 1128
10.228.3.8 CreateDynamicLine . . . 1129
10.228.3.9 DeleteDynamicLine . . . 1129
10.228.3.10 DetachObjects . . . 1129
10.228.3.11 DetachVisual . . . 1129
10.228.3.12 DetachVisual . . . 1129
10.228.3.13 DisableTrackVisual . . . 1129
10.228.3.14 EnableTrackVisual . . . 1129
10.228.3.15 Fini . . . 1130
10.228.3.16 GetAttachedObjectCount . . . 1130
10.228.3.17 GetBoundingBox . . . 1130
10.228.3.18 GetChild . . . 1130
10.228.3.19 GetChildCount . . . 1130
10.228.3.20 GetId . . . 1130
10.228.3.21 GetMaterialName . . . 1131
10.228.3.22 GetMeshName . . . 1131
10.228.3.23 GetName . . . 1131
10.228.3.24 GetNormalMap . . . 1131
10.228.3.25 GetParent . . . 1131
10.228.3.26 GetPose . . . 1131
10.228.3.27 GetPosition . . . 1132
10.228.3.28 GetRootVisual . . . 1132
10.228.3.29 GetRotation . . . 1132
10.228.3.30 GetScale . . . 1132
10.228.3.31 GetScene . . . 1132
10.228.3.32 GetSceneNode . . . 1132
10.228.3.33 GetShaderType . . . 1133
10.228.3.34 GetSubMeshName . . . 1133
10.228.3.35 GetTransparency . . . 1133
10.228.3.36 GetVisibilityFlags . . . 1133
10.228.3.37 GetVisible . . . 1133
10.228.3.38 GetWorldPose . . . 1134
10.228.3.39 HasAttachedObject . . . 1134
10.228.3.40 Init . . . 1134
10.228.3.41 InsertMesh . . . 1134
10.228.3.42 InsertMesh . . . 1134

10.228.3.43 IsPlane . . . 1134
10.228.3.44 IsStatic . . . 1135
10.228.3.45 Load . . . 1135
10.228.3.46 Load . . . 1135
10.228.3.47 LoadFromMsg . . . 1135
10.228.3.48 LoadPlugin . . . 1135
10.228.3.49 MakeStatic . . . 1136
10.228.3.50 MoveToPosition . . . 1136
10.228.3.51 MoveToPositions . . . 1136
10.228.3.52 RemovePlugin . . . 1136
10.228.3.53 SetAmbient . . . 1136
10.228.3.54 SetCastShadows . . . 1136
10.228.3.55 SetDiffuse . . . 1137
10.228.3.56 SetEmissive . . . 1137
10.228.3.57 SetHighlighted . . . 1137
10.228.3.58 SetId . . . 1137
10.228.3.59 SetMaterial . . . 1137
10.228.3.60 SetName . . . 1137
10.228.3.61 SetNormalMap . . . 1138
10.228.3.62 SetPose . . . 1138
10.228.3.63 SetPosition . . . 1138
10.228.3.64 SetRibbonTrail . . . 1138
10.228.3.65 SetRotation . . . 1138
10.228.3.66 SetScale . . . 1138
10.228.3.67 SetScene . . . 1139
10.228.3.68 SetShaderType . . . 1139
10.228.3.69 SetSkeletonPose . . . 1139
10.228.3.70 SetSpecular . . . 1139
10.228.3.71 SetTransparency . . . 1139
10.228.3.72 SetVisibilityFlags . . . 1139
10.228.3.73 SetVisible . . . 1140
10.228.3.74 SetWireframe . . . 1140
10.228.3.75 SetWorldPose . . . 1140
10.228.3.76 SetWorldPosition . . . 1140
10.228.3.77 SetWorldRotation . . . 1140
10.228.3.78 ShowBoundingBox . . . 1141
10.228.3.79 ShowCollision . . . 1141

10.228.3.6 ShowCOM . . . 1141
10.228.3.7 ShowJoints . . . 1141
10.228.3.8 ShowSkeleton . . . 1141
10.228.3.9 ToggleVisible . . . 1141
10.228.3.10 Update . . . 1141
10.228.3.11 UpdateFromMsg . . . 1142
10.228.4 Member Data Documentation . . . 1142
10.228.4.1 parent . . . 1142
10.228.4.2 scene . . . 1142
10.228.4.3 sceneNode . . . 1142
10.229 gazebo::VisualPlugin Class Reference . . . 1142
10.229.1 Detailed Description . . . 1143
10.229.2 Constructor & Destructor Documentation . . . 1143
10.229.2.1 VisualPlugin . . . 1143
10.229.3 Member Function Documentation . . . 1143
10.229.3.1 Init . . . 1143
10.229.3.2 Load . . . 1143
10.229.3.3 Reset . . . 1143
10.230 gazebo::rendering::WindowManager Class Reference . . . 1144
10.230.1 Detailed Description . . . 1144
10.230.2 Constructor & Destructor Documentation . . . 1144
10.230.2.1 WindowManager . . . 1144
10.230.2.2 ~WindowManager . . . 1144
10.230.3 Member Function Documentation . . . 1145
10.230.3.1 CreateWindow . . . 1145
10.230.3.2 Fini . . . 1145
10.230.3.3 GetAvgFPS . . . 1145
10.230.3.4 GetTriangleCount . . . 1145
10.230.3.5 GetWindow . . . 1145
10.230.3.6 Moved . . . 1146
10.230.3.7 Resize . . . 1146
10.230.3.8 SetCamera . . . 1146
10.231 gazebo::rendering::WireBox Class Reference . . . 1146
10.231.1 Detailed Description . . . 1147
10.231.2 Constructor & Destructor Documentation . . . 1147
10.231.2.1 WireBox . . . 1147
10.231.2.2 ~WireBox . . . 1147

10.231.3 Member Function Documentation . . . 1147
10.231.3.1 Init . . . 1147
10.231.3.2 SetVisible . . . 1147
10.232 gazebo::sensors::WirelessReceiver Class Reference . . . 1147
10.232.1 Detailed Description . . . 1149
10.232.2 Constructor & Destructor Documentation . . . 1149
10.232.2.1 WirelessReceiver . . . 1149
10.232.2.2 ~WirelessReceiver . . . 1149
10.232.3 Member Function Documentation . . . 1149
10.232.3.1 Fini . . . 1149
10.232.3.2 GetMaxFreqFiltered . . . 1149
10.232.3.3 GetMinFreqFiltered . . . 1149
10.232.3.4 GetSensitivity . . . 1149
10.232.3.5 Init . . . 1150
10.232.3.6 Load . . . 1150
10.233 gazebo::sensors::WirelessTransceiver Class Reference . . . 1150
10.233.1 Detailed Description . . . 1151
10.233.2 Constructor & Destructor Documentation . . . 1151
10.233.2.1 WirelessTransceiver . . . 1151
10.233.2.2 ~WirelessTransceiver . . . 1151
10.233.3 Member Function Documentation . . . 1152
10.233.3.1 Fini . . . 1152
10.233.3.2 GetGain . . . 1152
10.233.3.3 GetPower . . . 1152
10.233.3.4 GetTopic . . . 1152
10.233.3.5 Init . . . 1152
10.233.3.6 Load . . . 1152
10.233.4 Member Data Documentation . . . 1153
10.233.4.1 gain . . . 1153
10.233.4.2 parentEntity . . . 1153
10.233.4.3 power . . . 1153
10.233.4.4 pub . . . 1153
10.233.4.5 referencePose . . . 1153
10.234 gazebo::sensors::WirelessTransmitter Class Reference . . . 1153
10.234.1 Detailed Description . . . 1155
10.234.2 Constructor & Destructor Documentation . . . 1155
10.234.2.1 WirelessTransmitter . . . 1155

10.234.2.2 ~WirelessTransmitter . . . 1155
10.234.3 Member Function Documentation . . . 1155
10.234.3.1 GetESSID . . . 1155
10.234.3.2 GetFreq . . . 1155
10.234.3.3 GetSignalStrength . . . 1156
10.234.3.4 Init . . . 1156
10.234.3.5 Load . . . 1156
10.234.3.6 UpdateImpl . . . 1156
10.234.4 Member Data Documentation . . . 1156
10.234.4.1 freq . . . 1156
10.234.4.2 ModelStdDesv . . . 1156
10.234.4.3 NEmpty . . . 1157
10.234.4.4 NObstacle . . . 1157
10.235 gazebo::physics::World Class Reference . . . 1157
10.235.1 Detailed Description . . . 1160
10.235.2 Constructor & Destructor Documentation . . . 1160
10.235.2.1 World . . . 1160
10.235.2.2 ~World . . . 1160
10.235.3 Member Function Documentation . . . 1160
10.235.3.1 Clear . . . 1160
10.235.3.2 ClearModels . . . 1160
10.235.3.3 DisableAllModels . . . 1160
10.235.3.4 EnableAllModels . . . 1160
10.235.3.5 EnablePhysicsEngine . . . 1161
10.235.3.6 Fini . . . 1161
10.235.3.7 GetByName . . . 1161
10.235.3.8 GetEnablePhysicsEngine . . . 1161
10.235.3.9 GetEntity . . . 1161
10.235.3.10 GetEntityBelowPoint . . . 1162
10.235.3.11 GetModel . . . 1162
10.235.3.12 GetModel . . . 1162
10.235.3.13 GetModelBelowPoint . . . 1162
10.235.3.14 GetModelCount . . . 1163
10.235.3.15 GetModels . . . 1163
10.235.3.16 GetName . . . 1163
10.235.3.17 GetPauseTime . . . 1163
10.235.3.18 GetPhysicsEngine . . . 1163

10.235.3.19 GetRealTime . . . 1164
10.235.3.20 GetRunning . . . 1164
10.235.3.21 GetSelectedEntity . . . 1164
10.235.3.22 GetSetWorldPoseMutex . . . 1164
10.235.3.23 GetSimTime . . . 1164
10.235.3.24 GetSphericalCoordinates . . . 1164
10.235.3.25 GetStartTime . . . 1165
10.235.3.26 Init . . . 1165
10.235.3.27 InsertModelFile . . . 1165
10.235.3.28 InsertModelSDF . . . 1165
10.235.3.29 InsertModelString . . . 1165
10.235.3.30 IsLoaded . . . 1165
10.235.3.31 IsPaused . . . 1166
10.235.3.32 Load . . . 1166
10.235.3.33 LoadPlugin . . . 1166
10.235.3.34 PrintEntityTree . . . 1166
10.235.3.35 PublishModelPose . . . 1166
10.235.3.36 RemovePlugin . . . 1166
10.235.3.37 Reset . . . 1167
10.235.3.38 ResetEntities . . . 1167
10.235.3.39 ResetTime . . . 1167
10.235.3.40 Run . . . 1167
10.235.3.41 Save . . . 1167
10.235.3.42 SetPaused . . . 1167
10.235.3.43 SetSimTime . . . 1168
10.235.3.44 SetState . . . 1168
10.235.3.45 StepWorld . . . 1168
10.235.3.46 Stop . . . 1168
10.235.3.47 StripWorldName . . . 1168
10.235.3.48 UpdateStateSDF . . . 1168
10.235.4 Member Data Documentation . . . 1168
10.235.4.1 dirtyPoses . . . 1169
10.236 gazebo::WorldPlugin Class Reference . . . 1169
10.236.1 Detailed Description . . . 1169
10.236.2 Constructor & Destructor Documentation . . . 1170
10.236.2.1 WorldPlugin . . . 1170
10.236.2.2 ~WorldPlugin . . . 1170

10.236.3 Member Function Documentation . . . 1170
10.236.3.1 Init . . . 1170
10.236.3.2 Load . . . 1170
10.236.3.3 Reset . . . 1170
10.237 gazebo::physics::WorldState Class Reference . . . 1170
10.237.1 Detailed Description . . . 1172
10.237.2 Constructor & Destructor Documentation . . . 1172
10.237.2.1 WorldState . . . 1172
10.237.2.2 WorldState . . . 1172
10.237.2.3 WorldState . . . 1172
10.237.2.4 ~WorldState . . . 1173
10.237.3 Member Function Documentation . . . 1173
10.237.3.1 FillSDF . . . 1173
10.237.3.2 GetModelState . . . 1173
10.237.3.3 GetModelStateCount . . . 1173
10.237.3.4 GetModelStates . . . 1173
10.237.3.5 GetModelStates . . . 1174
10.237.3.6 HasModelState . . . 1174
10.237.3.7 IsZero . . . 1174
10.237.3.8 Load . . . 1174
10.237.3.9 Load . . . 1175
10.237.3.10 operator+ . . . 1175
10.237.3.11 operator- . . . 1175
10.237.3.12 operator= . . . 1175
10.237.3.13 SetRealTime . . . 1175
10.237.3.14 SetSimTime . . . 1176
10.237.3.15 SetWallTime . . . 1176
10.237.3.16 SetWorld . . . 1176
10.237.4 Friends And Related Function Documentation . . . 1176
10.237.4.1 operator<< . . . 1176
10.238 gazebo::rendering::WrenchVisual Class Reference . . . 1177
10.238.1 Detailed Description . . . 1177
10.238.2 Constructor & Destructor Documentation . . . 1177
10.238.2.1 WrenchVisual . . . 1178
10.238.2.2 ~WrenchVisual . . . 1178
10.238.3 Member Function Documentation . . . 1178
10.238.3.1 Load . . . 1178

10.238.3.2 SetEnabled . . . 1178

**11 File Documentation** **1179**

11.1 Actor.hh File Reference . . . 1179
11.2 Angle.hh File Reference . . . 1180
11.2.1 Macro Definition Documentation . . . 1182
11.2.1.1 GZ_DTOR . . . 1182
11.2.1.2 GZ_NORMALIZE . . . 1182
11.2.1.3 GZ_RTOD . . . 1182
11.3 Animation.hh File Reference . . . 1183
11.4 ArrowVisual.hh File Reference . . . 1184
11.5 Assert.hh File Reference . . . 1185
11.5.1 Macro Definition Documentation . . . 1186
11.5.1.1 GZ_ASSERT . . . 1186
11.6 AudioDecoder.hh File Reference . . . 1187
11.7 AxisVisual.hh File Reference . . . 1188
11.8 BallJoint.hh File Reference . . . 1188
11.9 Base.hh File Reference . . . 1189
11.10 Base64.hh File Reference . . . 1190
11.10.1 Function Documentation . . . 1192
11.10.1.1 Base64Decode . . . 1192
11.10.1.2 Base64Encode . . . 1192
11.11 Box.hh File Reference . . . 1192
11.12 BoxShape.hh File Reference . . . 1193
11.13 BVHLoader.hh File Reference . . . 1194
11.13.1 Macro Definition Documentation . . . 1196
11.13.1.1 X_POSITION . . . 1196
11.13.1.2 X_ROTATION . . . 1196
11.13.1.3 Y_POSITION . . . 1196
11.13.1.4 Y_ROTATION . . . 1196
11.13.1.5 Z_POSITION . . . 1196
11.13.1.6 Z_ROTATION . . . 1196
11.14 CallbackHelper.hh File Reference . . . 1196
11.15 Camera.hh File Reference . . . 1198
11.16 CameraSensor.hh File Reference . . . 1199
11.17 CameraVisual.hh File Reference . . . 1199
11.18 cegui.h File Reference . . . 1200

11.19ColladaLoader.hh File Reference . . . 1201
11.20Collision.hh File Reference . . . 1202
11.21CollisionState.hh File Reference . . . 1203
11.22Color.hh File Reference . . . 1203
11.23CommonIface.hh File Reference . . . 1204
11.24CommonTypes.hh File Reference . . . 1206
11.24.1 Macro Definition Documentation . . . 1208
11.24.1.1 GAZEBO_DEPRECATED . . . 1208
11.24.1.2 GAZEBO_FORCEINLINE . . . 1208
11.24.1.3 NULL . . . 1208
11.25COMVisual.hh File Reference . . . 1208
11.26Connection.hh File Reference . . . 1209
11.26.1 Macro Definition Documentation . . . 1211
11.26.1.1 HEADER_LENGTH . . . 1211
11.27ConnectionManager.hh File Reference . . . 1211
11.28Console.hh File Reference . . . 1213
11.29Contact.hh File Reference . . . 1214
11.29.1 Macro Definition Documentation . . . 1215
11.29.1.1 MAX_COLLIDE_RETURNS . . . 1215
11.29.1.2 MAX_CONTACT_JOINTS . . . 1215
11.30ContactManager.hh File Reference . . . 1216
11.31ContactSensor.hh File Reference . . . 1217
11.32ContactVisual.hh File Reference . . . 1217
11.33Conversions.hh File Reference . . . 1218
11.34CylinderShape.hh File Reference . . . 1219
11.35dart_inc.h File Reference . . . 1220
11.36DARTBallJoint.hh File Reference . . . 1221
11.37DARTBoxShape.hh File Reference . . . 1222
11.38DARTCollision.hh File Reference . . . 1222
11.39DARTCylinderShape.hh File Reference . . . 1223
11.40DARTHeightmapShape.hh File Reference . . . 1224
11.41DARTHinge2Joint.hh File Reference . . . 1224
11.42DARTHingeJoint.hh File Reference . . . 1225
11.43DARTJoint.hh File Reference . . . 1226
11.44DARTLink.hh File Reference . . . 1226
11.45DARTMeshShape.hh File Reference . . . 1227
11.46DARTModel.hh File Reference . . . 1227

11.47DARTMultiRayShape.hh File Reference . . . 1228
11.48DARTPhysics.hh File Reference . . . 1229
11.49DARTPlaneShape.hh File Reference . . . 1229
11.50DARTRayShape.hh File Reference . . . 1230
11.51DARTScrewJoint.hh File Reference . . . 1230
11.52DARTSliderJoint.hh File Reference . . . 1231
11.53DARTSphereShape.hh File Reference . . . 1231
11.54DARTTypes.hh File Reference . . . 1232
11.54.1 Detailed Description . . . 1233
11.55DARTUniversalJoint.hh File Reference . . . 1233
11.56DepthCamera.hh File Reference . . . 1234
11.57DepthCameraSensor.hh File Reference . . . 1234
11.58Diagnostics.hh File Reference . . . 1235
11.59DynamicLines.hh File Reference . . . 1236
11.60DynamicRenderable.hh File Reference . . . 1237
11.61Entity.hh File Reference . . . 1237
11.62Event.hh File Reference . . . 1238
11.63Events.hh File Reference . . . 1239
11.64Exception.hh File Reference . . . 1240
11.65ForceTorqueSensor.hh File Reference . . . 1242
11.66FPSViewController.hh File Reference . . . 1242
11.67gazebo.hh File Reference . . . 1243
11.68gazebo_core.hh File Reference . . . 1244
11.69GazeboGenerator.hh File Reference . . . 1245
11.70GpsSensor.hh File Reference . . . 1246
11.71GpuLaser.hh File Reference . . . 1247
11.72GpuRaySensor.hh File Reference . . . 1247
11.73Grid.hh File Reference . . . 1248
11.74Gripper.hh File Reference . . . 1249
11.75GUIOverlay.hh File Reference . . . 1250
11.76Heightmap.hh File Reference . . . 1250
11.77HeightmapShape.hh File Reference . . . 1251
11.78Helpers.hh File Reference . . . 1252
11.78.1 Macro Definition Documentation . . . 1254
11.78.1.1 GZ_DBL_MAX . . . 1254
11.78.1.2 GZ_DBL_MIN . . . 1254
11.78.1.3 GZ_FLT_MAX . . . 1254

11.78.1.4 GZ_FLT_MIN . . . 1254
11.78.1.5 GZ_UINT32_MAX . . . 1255
11.78.1.6 GZ_UINT32_MIN . . . 1255
11.79 Hinge2Joint.hh File Reference . . . 1255
11.80 HingeJoint.hh File Reference . . . 1256
11.81 Image.hh File Reference . . . 1257
11.82 ImuSensor.hh File Reference . . . 1258
11.83 Inertial.hh File Reference . . . 1258
11.84 IOManager.hh File Reference . . . 1259
11.85 Joint.hh File Reference . . . 1261
11.85.1 Macro Definition Documentation . . . 1262
11.85.1.1 MAX_JOINT_AXIS . . . 1262
11.86 Joint_TEST.hh File Reference . . . 1262
11.86.1 Typedef Documentation . . . 1263
11.86.1.1 std_string2 . . . 1263
11.87 JointController.hh File Reference . . . 1263
11.88 JointState.hh File Reference . . . 1264
11.89 JointVisual.hh File Reference . . . 1265
11.90 JointWrench.hh File Reference . . . 1265
11.91 KeyEvent.hh File Reference . . . 1266
11.92 KeyFrame.hh File Reference . . . 1267
11.93 LaserVisual.hh File Reference . . . 1268
11.94 Light.hh File Reference . . . 1269
11.95 Link.hh File Reference . . . 1270
11.96 LinkState.hh File Reference . . . 1271
11.97 LogPlay.hh File Reference . . . 1273
11.98 LogRecord.hh File Reference . . . 1273
11.98.1 Macro Definition Documentation . . . 1274
11.98.1.1 GZ_LOG_VERSION . . . 1274
11.99 mainpage.html File Reference . . . 1274
11.100 MapShape.hh File Reference . . . 1274
11.101 Master.hh File Reference . . . 1275
11.102 Material.hh File Reference . . . 1276
11.103 Material.hh File Reference . . . 1278
11.104 MathTypes.hh File Reference . . . 1278
11.104.1 Detailed Description . . . 1278
11.105 Matrix3.hh File Reference . . . 1278

11.106 Matrix4.hh File Reference . . . 1279
11.107 Mesh.hh File Reference . . . 1280
11.108 MeshCSG.hh File Reference . . . 1282
11.108.1 Typedef Documentation . . . 1283
11.108.1.1 GPtrArray . . . 1283
11.108.1.2 GtsSurface . . . 1283
11.109 MeshLoader.hh File Reference . . . 1284
11.110 MeshManager.hh File Reference . . . 1285
11.111 MeshShape.hh File Reference . . . 1286
11.112 Model.hh File Reference . . . 1287
11.113 ModelDatabase.hh File Reference . . . 1289
11.113.1 Macro Definition Documentation . . . 1289
11.113.1.1 GZ_MODEL_DB_MANIFEST_FILENAME . . . 1289
11.113.1.2 GZ_MODEL_MANIFEST_FILENAME . . . 1290
11.114 ModelState.hh File Reference . . . 1290
11.115 MouseEvent.hh File Reference . . . 1291
11.116 MovableText.hh File Reference . . . 1293
11.117 MsgFactory.hh File Reference . . . 1293
11.118 msgs.hh File Reference . . . 1294
11.119 MultiCameraSensor.hh File Reference . . . 1297
11.120 MultiRayShape.hh File Reference . . . 1298
11.121 Node.hh File Reference . . . 1299
11.122 Noise.hh File Reference . . . 1300
11.123 ogre_gazebo.h File Reference . . . 1301
11.124 OpenAL.hh File Reference . . . 1302
11.125 OrbitViewController.hh File Reference . . . 1303
11.126 PhysicsEngine.hh File Reference . . . 1303
11.127 PhysicsFactory.hh File Reference . . . 1304
11.128 PhysicsIface.hh File Reference . . . 1306
11.129 PhysicsTypes.hh File Reference . . . 1308
11.129.1 Detailed Description . . . 1310
11.129.2 Macro Definition Documentation . . . 1310
11.129.2.1 GZ_ALL_COLLIDE . . . 1310
11.129.2.2 GZ_FIXED_COLLIDE . . . 1310
11.129.2.3 GZ_GHOST_COLLIDE . . . 1310
11.129.2.4 GZ_NONE_COLLIDE . . . 1310
11.129.2.5 GZ_SENSOR_COLLIDE . . . 1311

11.130 PID.hh File Reference . . . 1311
11.131 Plane.hh File Reference . . . 1312
11.132 PlaneShape.hh File Reference . . . 1312
11.133 Plugin.hh File Reference . . . 1313
11.133.1 Macro Definition Documentation . . . 1315
11.133.1.1 GZ_REGISTER_MODEL_PLUGIN . . . 1315
11.133.1.2 GZ_REGISTER_SENSOR_PLUGIN . . . 1316
11.133.1.3 GZ_REGISTER_SYSTEM_PLUGIN . . . 1316
11.133.1.4 GZ_REGISTER_VISUAL_PLUGIN . . . 1316
11.133.1.5 GZ_REGISTER_WORLD_PLUGIN . . . 1317
11.134 Pose.hh File Reference . . . 1317
11.135 Projector.hh File Reference . . . 1318
11.136 Publication.hh File Reference . . . 1318
11.137 PublicationTransport.hh File Reference . . . 1321
11.138 Publisher.hh File Reference . . . 1323
11.139 Quaternion.hh File Reference . . . 1325
11.140 Rand.hh File Reference . . . 1326
11.141 RaySensor.hh File Reference . . . 1327
11.142 RayShape.hh File Reference . . . 1328
11.143 RenderEngine.hh File Reference . . . 1329
11.144 RenderEvents.hh File Reference . . . 1330
11.145 RenderingIface.hh File Reference . . . 1330
11.146 RenderTypes.hh File Reference . . . 1332
11.146.1 Macro Definition Documentation . . . 1333
11.146.1.1 GZ_VISIBILITY_ALL . . . 1333
11.146.1.2 GZ_VISIBILITY_GUI . . . 1334
11.146.1.3 GZ_VISIBILITY_SELECTABLE . . . 1334
11.146.1.4 GZ_VISIBILITY_SELECTION . . . 1334
11.147 RFIDSensor.hh File Reference . . . 1334
11.148 RFIDTag.hh File Reference . . . 1335
11.149 RFIDTagVisual.hh File Reference . . . 1335
11.150 RFIDVisual.hh File Reference . . . 1336
11.151 Road.hh File Reference . . . 1337
11.152 Road2d.hh File Reference . . . 1338
11.153 RotationSpline.hh File Reference . . . 1338
11.154 RTShaderSystem.hh File Reference . . . 1340
11.155 Scene.hh File Reference . . . 1340

11.155 ScrewJoint.hh File Reference . . . 1342
11.157 SelectionObj.hh File Reference . . . 1343
11.158 Sensor.hh File Reference . . . 1343
11.159 SensorFactory.hh File Reference . . . 1344
11.160 SensorManager.hh File Reference . . . 1345
11.161 SensorsIface.hh File Reference . . . 1346
11.162 SensorTypes.hh File Reference . . . 1348
11.162.1 Detailed Description . . . 1350
11.163 Server.hh File Reference . . . 1350
11.164 Shape.hh File Reference . . . 1351
11.165 Simbody_inc.h File Reference . . . 1352
11.166 SimbodyBallJoint.hh File Reference . . . 1353
11.167 SimbodyBoxShape.hh File Reference . . . 1354
11.168 SimbodyCollision.hh File Reference . . . 1354
11.169 SimbodyCylinderShape.hh File Reference . . . 1355
11.170 SimbodyHeightmapShape.hh File Reference . . . 1355
11.171 SimbodyHinge2Joint.hh File Reference . . . 1356
11.172 SimbodyHingeJoint.hh File Reference . . . 1357
11.173 SimbodyJoint.hh File Reference . . . 1357
11.174 SimbodyLink.hh File Reference . . . 1358
11.175 SimbodyMeshShape.hh File Reference . . . 1359
11.176 SimbodyModel.hh File Reference . . . 1359
11.177 SimbodyMultiRayShape.hh File Reference . . . 1360
11.178 SimbodyPhysics.hh File Reference . . . 1360
11.179 SimbodyPlaneShape.hh File Reference . . . 1361
11.180 SimbodyRayShape.hh File Reference . . . 1362
11.181 SimbodyScrewJoint.hh File Reference . . . 1363
11.182 SimbodySliderJoint.hh File Reference . . . 1363
11.183 SimbodySphereShape.hh File Reference . . . 1364
11.184 SimbodyTypes.hh File Reference . . . 1364
11.184.1 Detailed Description . . . 1366
11.185 SimbodyUniversalJoint.hh File Reference . . . 1366
11.186 SingletonT.hh File Reference . . . 1366
11.187 Skeleton.hh File Reference . . . 1367
11.188 SkeletonAnimation.hh File Reference . . . 1369
11.189 SliderJoint.hh File Reference . . . 1370
11.190 SonarSensor.hh File Reference . . . 1371

11.191 SonarVisual.hh File Reference . . . 1372
11.192 SphereShape.hh File Reference . . . 1373
11.193 SphericalCoordinates.hh File Reference . . . 1374
11.194 Spline.hh File Reference . . . 1375
11.195 State.hh File Reference . . . 1377
11.196 STLLoader.hh File Reference . . . 1378
11.196.1 Macro Definition Documentation . . . 1379
11.196.1.1 COR3_MAX . . . 1379
11.196.1.2 FACE_MAX . . . 1379
11.196.1.3 LINE_MAX_LEN . . . 1379
11.196.1.4 ORDER_MAX . . . 1379
11.197 SubscribeOptions.hh File Reference . . . 1379
11.198 Subscriber.hh File Reference . . . 1381
11.199 SubscriptionTransport.hh File Reference . . . 1383
11.200 SurfaceParams.hh File Reference . . . 1385
11.201 SystemPaths.hh File Reference . . . 1386
11.201.1 Macro Definition Documentation . . . 1387
11.201.1.1 GetCurrentDir . . . 1387
11.201.1.2 LINUX . . . 1387
11.202 Time.hh File Reference . . . 1387
11.203 Timer.hh File Reference . . . 1388
11.204 TopicManager.hh File Reference . . . 1389
11.205 TransmitterVisual.hh File Reference . . . 1391
11.206 TransportIface.hh File Reference . . . 1392
11.207 TransportTypes.hh File Reference . . . 1394
11.207.1 Detailed Description . . . 1395
11.208 UniversalJoint.hh File Reference . . . 1395
11.209 UpdateInfo.hh File Reference . . . 1396
11.210 UserCamera.hh File Reference . . . 1397
11.211 UtilTypes.hh File Reference . . . 1398
11.212 Vector2d.hh File Reference . . . 1399
11.213 Vector2i.hh File Reference . . . 1400
11.214 Vector3.hh File Reference . . . 1401
11.215 Vector4.hh File Reference . . . 1401
11.216 Video.hh File Reference . . . 1403
11.217 VideoVisual.hh File Reference . . . 1404
11.218 ViewController.hh File Reference . . . 1405

11.219Visual.hh File Reference . . . . . 1406
11.220WindowManager.hh File Reference . . . . . 1407
11.221WireBox.hh File Reference . . . . . 1408
11.222WirelessReceiver.hh File Reference . . . . . 1409
11.223WirelessTransceiver.hh File Reference . . . . . 1409
11.224WirelessTransmitter.hh File Reference . . . . . 1410
11.225World.hh File Reference . . . . . 1411
11.226WorldState.hh File Reference . . . . . 1412
11.227WrenchVisual.hh File Reference . . . . . 1414

**Index** **1414**

# Chapter 1

# Gazebo API Reference

This documentation provides useful information about the Gazebo API. The code reference is divided into the groups below. Should you find problems with this documentation - typos, unclear phrases, or insufficient detail - please create a `new bitbucket issue`. Include sufficient detail to quickly locate the problematic documentation, and set the issue's fields accordingly: Assignee - blank; Kind - bug; Priority - minor; Version - blank.

**Class** `List` - Index of all classes in Gazebo, organized alphabetically

`Hierarchy` - Index of classes, organized hierachically according to their inheritance

`Modules Common`: Classes and files used ubiquitously across Gazebo

`Events`: For creating and destroying Gazebo events

`Math`: A set of classes that encapsulate math related properties and functions.

`Messages`: All messages and helper functions.

`Physics`: Classes for physics and dynamics

`Rendering`: A set of rendering related class, functions, and definitions.

`Sensors`: A set of sensor classes, functions, and definitions.

`Transport`: Handles transportation of messages.

**Links** `Website`: The main gazebo website, which contains news, downloads, and contact information.

`Wiki`: A collection of user supported documentation.

`Tutorials`: Tutorials that describe how to use Gazebo and implement your own simulations.

`Download`: How to download and install Gazebo

# Chapter 2

# Todo List

**Member gazebo::physics::Joint::GetForce (**p. **505) (unsigned int _index)**

: not yet implemented. Get external forces applied at this Joint. Note that the unit of force should be consistent with the rest of the simulation scales.

**Member gazebo::sensors::CameraSensor::GetTopic (**p. **215) () const**

to be implemented

**Class gazebo::SystemPlugin (**p. **1030)**

how to make doxygen reference to the file gazebo.cc::g_plugins?

# Chapter 3

# Module Index

## 3.1 Modules

Here is a list of all modules:


Common . . . . . 33
Events . . . . . 45
Math . . . . . 48
Messages . . . . . 54
Classes for physics and dynamics . . . . . 65
    DART Physics . . . . . 73
    Bullet Physics . . . . . 76
    Simbody Physics . . . . . 78
Rendering . . . . . 80
Sensors . . . . . 87
Transport . . . . . 92
Utility . . . . . 98

# Chapter 4

# Namespace Index

## 4.1 Namespace List

Here is a list of all namespaces with brief descriptions:


**boost** . . . 99
**gazebo**
Forward declarations for the common classes . . . 99
**gazebo::common**
Common namespace . . . 101
**gazebo::event**
**Event** (p. 390) namespace . . . 105
**gazebo::math**
Math namespace . . . 105
**gazebo::msgs**
Messages namespace . . . 108
**gazebo::physics**
Namespace for physics . . . 110
**gazebo::rendering**
Rendering namespace . . . 118
**gazebo::sensors**
Sensors namespace . . . 122
**gazebo::transport** . . . 127
**gazebo::util** . . . 129
**google** . . . 130
**google::protobuf** . . . 130
**google::protobuf::compiler** . . . 130
**google::protobuf::compiler::cpp** . . . 130
**Ogre** . . . 130
**ogre** . . . 130
**SimTK** . . . 130
**SkyX** . . . 130

# Chapter 5

# Hierarchical Index

## 5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:


gazebo::math::Angle . . . 137
gazebo::common::Animation . . . 145
  gazebo::common::NumericAnimation . . . 692
  gazebo::common::PoseAnimation . . . 743
gazebo::common::AudioDecoder . . . 153
gazebo::math::Box . . . 172
gazebo::common::BVHLoader . . . 179
gazebo::transport::CallbackHelper . . . 180
  gazebo::transport::CallbackHelperT< M > . . . 184
  gazebo::transport::RawCallbackHelper . . . 776
  gazebo::transport::SubscriptionTransport . . . 1018
CodeGenerator
  google::protobuf::compiler::cpp::GazeboGenerator . . . 425
gazebo::common::Color . . . 233
gazebo::event::Connection . . . 246
gazebo::physics::Contact . . . 260
gazebo::physics::ContactManager . . . 263
gazebo::physics::ContactPublisher . . . 266
gazebo::rendering::Conversions . . . 273
gazebo::physics::DARTTypes . . . 351
enable_shared_from_this
  gazebo::physics::Base . . . 159
    gazebo::physics::Entity . . . 379
      gazebo::physics::Collision . . . 220
        gazebo::physics::DARTCollision . . . 286
        gazebo::physics::SimbodyCollision . . . 872
      gazebo::physics::Link . . . 542
        gazebo::physics::DARTLink . . . 313
        gazebo::physics::SimbodyLink . . . 900
      gazebo::physics::Model . . . 624
        gazebo::physics::Actor . . . 131
        gazebo::physics::DARTModel . . . 326
        gazebo::physics::SimbodyModel . . . 912

gazebo::physics::Joint . . . . . 496
gazebo::physics::DARTJoint . . . . . 304
gazebo::physics::BallJoint< DARTJoint > . . . . . 157
gazebo::physics::DARTBallJoint . . . . . 279
gazebo::physics::Hinge2Joint< DARTJoint > . . . . . 471
gazebo::physics::DARTHinge2Joint . . . . . 294
gazebo::physics::HingeJoint< DARTJoint > . . . . . 472
gazebo::physics::DARTHingeJoint . . . . . 299
gazebo::physics::ScrewJoint< DARTJoint > . . . . . 832
gazebo::physics::DARTScrewJoint . . . . . 339
gazebo::physics::SliderJoint< DARTJoint > . . . . . 973
gazebo::physics::DARTSliderJoint . . . . . 345
gazebo::physics::UniversalJoint< DARTJoint > . . . . . 1064
gazebo::physics::DARTUniversalJoint . . . . . 352
gazebo::physics::SimbodyJoint . . . . . 892
gazebo::physics::BallJoint< SimbodyJoint > . . . . . 157
gazebo::physics::SimbodyBallJoint . . . . . 865
gazebo::physics::Hinge2Joint< SimbodyJoint > . . . . . 471
gazebo::physics::SimbodyHinge2Joint . . . . . 879
gazebo::physics::HingeJoint< SimbodyJoint > . . . . . 472
gazebo::physics::SimbodyHingeJoint . . . . . 886
gazebo::physics::ScrewJoint< SimbodyJoint > . . . . . 832
gazebo::physics::SimbodyScrewJoint . . . . . 929
gazebo::physics::SliderJoint< SimbodyJoint > . . . . . 973
gazebo::physics::SimbodySliderJoint . . . . . 936
gazebo::physics::UniversalJoint< SimbodyJoint > . . . . . 1064
gazebo::physics::SimbodyUniversalJoint . . . . . 944
gazebo::physics::Road . . . . . 804
gazebo::physics::Shape . . . . . 861
gazebo::physics::BoxShape . . . . . 176
gazebo::physics::DARTBoxShape . . . . . 284
gazebo::physics::SimbodyBoxShape . . . . . 871
gazebo::physics::CylinderShape . . . . . 275
gazebo::physics::DARTCylinderShape . . . . . 290
gazebo::physics::SimbodyCylinderShape . . . . . 875
gazebo::physics::HeightmapShape . . . . . 465
gazebo::physics::DARTHeightmapShape . . . . . 292
gazebo::physics::SimbodyHeightmapShape . . . . . 877
gazebo::physics::MapShape . . . . . 579
gazebo::physics::MeshShape . . . . . 621
gazebo::physics::DARTMeshShape . . . . . 324
gazebo::physics::SimbodyMeshShape . . . . . 910
gazebo::physics::MultiRayShape . . . . . 664
gazebo::physics::DARTMultiRayShape . . . . . 328
gazebo::physics::SimbodyMultiRayShape . . . . . 913
gazebo::physics::PlaneShape . . . . . 728
gazebo::physics::DARTPlaneShape . . . . . 336
gazebo::physics::SimbodyPlaneShape . . . . . 924
gazebo::physics::RayShape . . . . . 786
gazebo::physics::DARTRayShape . . . . . 338
gazebo::physics::SimbodyRayShape . . . . . 926

gazebo::physics::SphereShape . . . 986
gazebo::physics::DARTSphereShape . . . 350
gazebo::physics::SimbodySphereShape . . . 942
gazebo::physics::World . . . 1157
gazebo::rendering::Camera . . . 186
gazebo::rendering::DepthCamera . . . 357
gazebo::rendering::GpuLaser . . . 429
gazebo::rendering::UserCamera . . . 1066
gazebo::rendering::Scene . . . 814
gazebo::rendering::Visual . . . 1121
gazebo::rendering::ArrowVisual . . . 149
gazebo::rendering::AxisVisual . . . 155
gazebo::rendering::CameraVisual . . . 217
gazebo::rendering::COMVisual . . . 244
gazebo::rendering::ContactVisual . . . 272
gazebo::rendering::JointVisual . . . 527
gazebo::rendering::LaserVisual . . . 534
gazebo::rendering::RFIDTagVisual . . . 801
gazebo::rendering::RFIDVisual . . . 802
gazebo::rendering::SelectionObj . . . 835
gazebo::rendering::SonarVisual . . . 982
gazebo::rendering::TransmitterVisual . . . 1062
gazebo::rendering::VideoVisual . . . 1116
gazebo::rendering::WrenchVisual . . . 1177
gazebo::sensors::Sensor . . . 837
gazebo::sensors::CameraSensor . . . 213
gazebo::sensors::ContactSensor . . . 267
gazebo::sensors::DepthCameraSensor . . . 362
gazebo::sensors::ForceTorqueSensor . . . 418
gazebo::sensors::GpsSensor . . . 426
gazebo::sensors::GpuRaySensor . . . 438
gazebo::sensors::ImuSensor . . . 480
gazebo::sensors::MultiCameraSensor . . . 660
gazebo::sensors::RaySensor . . . 779
gazebo::sensors::RFIDSensor . . . 796
gazebo::sensors::RFIDTag . . . 798
gazebo::sensors::SonarSensor . . . 977
gazebo::sensors::WirelessTransceiver . . . 1150
gazebo::sensors::WirelessReceiver . . . 1147
gazebo::sensors::WirelessTransmitter . . . 1153
gazebo::transport::Connection . . . 247
gazebo::transport::Node . . . 672
gazebo::event::Event . . . 390
gazebo::event::EventT< void() > . . . 407
gazebo::event::EventT< void(bool) > . . . 407
gazebo::event::EventT< void(const common::UpdateInfo &) > . . . 407
gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &) > . . . 407
gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format) > . . . 407
gazebo::event::EventT< void(const std::string &) > . . . 407
gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &) > . . . 407
gazebo::event::EventT< void(msgs::SonarStamped) > . . . 407
gazebo::event::EventT< void(msgs::WrenchStamped) > . . . 407

gazebo::event::EventT< void(std::string)> . . . 407
gazebo::event::EventT< void(std::string, std::string)> . . . 407
gazebo::event::EventT< T > . . . 407
gazebo::event::Events . . . 393
gazebo::rendering::Events . . . 405
gazebo::common::Exception . . . 416
gazebo::common::InternalError . . . 493
gazebo::common::AssertionInternalError . . . 151
gazebo::rendering::Grid . . . 450
gazebo::physics::Gripper . . . 453
gazebo::rendering::GUIOverlay . . . 455
gazebo::rendering::Heightmap . . . 460
gazebo::common::Image . . . 474
gazebo::physics::Inertial . . . 483
gazebo::transport::IOManager . . . 494
gazebo::physics::JointController . . . 521
gazebo::physics::JointWrench . . . 529
gazebo::common::KeyEvent . . . 531
gazebo::common::KeyFrame . . . 532
gazebo::common::NumericKeyFrame . . . 693
gazebo::common::PoseKeyFrame . . . 746
gazebo::rendering::Light . . . 535
Logplay . . . 573
gazebo::Master . . . 583
gazebo::common::Material . . . 585
gazebo::math::Matrix3 . . . 594
gazebo::math::Matrix4 . . . 599
gazebo::common::Mesh . . . 606
gazebo::common::MeshCSG . . . 613
gazebo::common::MeshLoader . . . 614
gazebo::common::ColladaLoader . . . 218
gazebo::common::STLLoader . . . 1002
gazebo::common::MouseEvent . . . 649
MovableObject
gazebo::rendering::MovableText . . . 652
gazebo::msgs::MsgFactory . . . 658
gazebo::common::NodeAnimation . . . 679
gazebo::common::NodeAssignment . . . 683
gazebo::common::NodeTransform . . . 684
gazebo::sensors::Noise . . . 689
gazebo::util::OpenALSink . . . 697
gazebo::util::OpenALSource . . . 698
PageProvider
gazebo::rendering::DummyPageProvider . . . 370
gazebo::common::ParamT< T > . . . 706
gazebo::physics::PhysicsEngine . . . 706
gazebo::physics::DARTPhysics . . . 330
gazebo::physics::SimbodyPhysics . . . 915
gazebo::physics::PhysicsFactory . . . 720
gazebo::common::PID . . . 722
gazebo::math::Plane . . . 726
gazebo::PluginT< T > . . . 732
gazebo::PluginT< ModelPlugin > . . . 732

gazebo::ModelPlugin . . . 639
gazebo::PluginT< SensorPlugin > . . . 732
gazebo::SensorPlugin . . . 853
gazebo::PluginT< SystemPlugin > . . . 732
gazebo::SystemPlugin . . . 1030
gazebo::PluginT< VisualPlugin > . . . 732
gazebo::VisualPlugin . . . 1142
gazebo::PluginT< WorldPlugin > . . . 732
gazebo::WorldPlugin . . . 1169
gazebo::math::Pose . . . 734
gazebo::rendering::Projector . . . 748
gazebo::transport::Publication . . . 750
gazebo::transport::PublicationTransport . . . 755
gazebo::transport::Publisher . . . 757
QuadNode . . . 760
gazebo::math::Quaternion . . . 761
gazebo::math::Rand . . . 775
Renderable
gazebo::rendering::MovableText . . . 652
RenderObjectListener
gazebo::rendering::GpuLaser . . . 429
Road . . . 804
gazebo::rendering::Road2d . . . 806
gazebo::math::RotationSpline . . . 806
SensorFactor . . . 848
gazebo::sensors::SensorFactory . . . 848
gazebo::Server . . . 855
ServerFixture
Joint_TEST . . . 516
ShaderHelperCg
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg . . . 857
ShaderHelperGLSL
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL . . . 859
SimpleRenderable
gazebo::rendering::DynamicRenderable . . . 375
gazebo::rendering::DynamicLines . . . 371
SingletonT< T > . . . 951
gazebo::common::Console . . . 259
gazebo::common::MeshManager . . . 616
gazebo::common::ModelDatabase . . . 638
gazebo::common::SystemPaths . . . 1025
gazebo::rendering::RenderEngine . . . 791
gazebo::rendering::RTShaderSystem . . . 810
gazebo::sensors::SensorManager . . . 850
gazebo::transport::ConnectionManager . . . 254
gazebo::transport::TopicManager . . . 1055
gazebo::util::DiagnosticManager . . . 365
gazebo::util::LogPlay . . . 570
gazebo::util::LogRecord . . . 573
gazebo::util::OpenAL . . . 695
SingletonT< ConnectionManager > . . . 951
SingletonT< Console > . . . 951

SingletonT< DiagnosticManager > . . . 951
SingletonT< LogPlay > . . . 951
SingletonT< LogRecord > . . . 951
SingletonT< MeshManager > . . . 951
SingletonT< ModelDatabase > . . . 951
SingletonT< OpenAL > . . . 951
SingletonT< RenderEngine > . . . 951
SingletonT< RTShaderSystem > . . . 951
SingletonT< SensorManager > . . . 951
SingletonT< SystemPaths > . . . 951
SingletonT< TopicManager > . . . 951
gazebo::common::Skeleton . . . 953
gazebo::common::SkeletonAnimation . . . 960
gazebo::common::SkeletonNode . . . 964
SM2Profile
  gazebo::rendering::GzTerrainMatGen::SM2Profile . . . 975
Joint_TEST::SpawnJointOptions . . . 983
gazebo::common::SphericalCoordinates . . . 988
gazebo::math::Spline . . . 993
gazebo::physics::State . . . 998
  gazebo::physics::CollisionState . . . 229
  gazebo::physics::JointState . . . 522
  gazebo::physics::LinkState . . . 563
  gazebo::physics::ModelState . . . 641
  gazebo::physics::WorldState . . . 1170
gazebo::common::SubMesh . . . 1004
gazebo::transport::SubscribeOptions . . . 1014
gazebo::transport::Subscriber . . . 1016
gazebo::physics::SurfaceParams . . . 1020
TerrainMaterialGeneratorA
  gazebo::rendering::GzTerrainMatGen . . . 459
gazebo::common::Time . . . 1031
gazebo::common::Timer . . . 1053
  gazebo::util::DiagnosticTimer . . . 368
gazebo::physics::TrajectoryInfo . . . 1061
gazebo::common::UpdateInfo . . . 1065
gazebo::math::Vector2d . . . 1074
gazebo::math::Vector2i . . . 1083
gazebo::math::Vector3 . . . 1091
gazebo::math::Vector4 . . . 1105
gazebo::common::Video . . . 1115
gazebo::rendering::ViewController . . . 1118
  gazebo::rendering::FPSViewController . . . 422
  gazebo::rendering::OrbitViewController . . . 703
gazebo::rendering::WindowManager . . . 1144
gazebo::rendering::WireBox . . . 1146
WithParamInterface
  Joint_TEST . . . 516
T
  gazebo::physics::BallJoint< T > . . . 157
  gazebo::physics::Hinge2Joint< T > . . . 471
  gazebo::physics::HingeJoint< T > . . . 472
  gazebo::physics::ScrewJoint< T > . . . 832

gazebo::physics::SliderJoint< T > . . . . . 973
gazebo::physics::UniversalJoint< T > . . . . . 1064

# Chapter 6

# Class Index

## 6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


**gazebo::physics::Actor**
**Actor** (p. 131) class enables GPU based mesh model / skeleton scriptable animation . . . . . 131
**gazebo::math::Angle**
An angle and related functions . . . . . 137
**gazebo::common::Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes . . . . . 145
**gazebo::rendering::ArrowVisual**
Basic arrow visualization . . . . . 149
**gazebo::common::AssertionInternalError**
Class for generating Exceptions which come from gazebo assertions . . . . . 151
**gazebo::common::AudioDecoder**
An audio decoder based on FFMPEG . . . . . 153
**gazebo::rendering::AxisVisual**
Basic axis visualization . . . . . 155
**gazebo::physics::BallJoint**< **T** >
**Base** (p. 159) class for a ball joint . . . . . 157
**gazebo::physics::Base**
**Base** (p. 159) class for most physics classes . . . . . 159
**gazebo::math::Box**
Mathematical representation of a box and related functions . . . . . 172
**gazebo::physics::BoxShape**
Box geometry primitive . . . . . 176
**gazebo::common::BVHLoader**
Handles loading BVH animation files . . . . . 179
**gazebo::transport::CallbackHelper**
A helper class to handle callbacks when messages arrive . . . . . 180
**gazebo::transport::CallbackHelperT**< **M** >
Callback helper Template . . . . . 184
**gazebo::rendering::Camera**
Basic camera sensor . . . . . 186
**gazebo::sensors::CameraSensor**
Basic camera sensor . . . . . 213

**gazebo::rendering::CameraVisual**
Basic camera visualization . . . 217
**gazebo::common::ColladaLoader**
Class used to load Collada mesh files . . . 218
**gazebo::physics::Collision**
**Base** (p. 159) class for all collision entities . . . 220
**gazebo::physics::CollisionState**
Store state information of a **physics::Collision** (p. 220) object . . . 229
**gazebo::common::Color**
Defines a color . . . 233
**gazebo::rendering::COMVisual**
Basic Center of Mass visualization . . . 244
**gazebo::event::Connection**
A class that encapsulates a connection . . . 246
**gazebo::transport::Connection**
Single TCP/IP connection manager . . . 247
**gazebo::transport::ConnectionManager**
Manager of connections . . . 254
**gazebo::common::Console**
Message, error, warning functionality . . . 259
**gazebo::physics::Contact**
A contact between two collisions . . . 260
**gazebo::physics::ContactManager**
Aggregates all the contact information generated by the collision detection engine . . . 263
**gazebo::physics::ContactPublisher**
A custom contact publisher created for each contact filter in the **Contact** (p. 260) Manager . . . 266
**gazebo::sensors::ContactSensor**
Contact sensor . . . 267
**gazebo::rendering::ContactVisual**
Contact visualization . . . 272
**gazebo::rendering::Conversions**
**Conversions** (p. 273) **Conversions.hh** (p. 1218) **rendering/Conversions.hh** (p. 1218) . . . 273
**gazebo::physics::CylinderShape**
Cylinder collision . . . 275
**gazebo::physics::DARTBallJoint**
An **DARTBallJoint** (p. 279) . . . 279
**gazebo::physics::DARTBoxShape**
DART Box shape . . . 284
**gazebo::physics::DARTCollision**
**Base** (p. 159) class for all DART collisions . . . 286
**gazebo::physics::DARTCylinderShape**
DART cylinder shape . . . 290
**gazebo::physics::DARTHeightmapShape**
DART Height map collision . . . 292
**gazebo::physics::DARTHinge2Joint**
A two axis hinge joint . . . 294
**gazebo::physics::DARTHingeJoint**
A single axis hinge joint . . . 299
**gazebo::physics::DARTJoint**
DART joint interface . . . 304
**gazebo::physics::DARTLink**
DART **Link** (p. 542) class . . . 313
**gazebo::physics::DARTMeshShape**
Triangle mesh collision . . . 324

**gazebo::physics::DARTModel**
DART model class . . . 326
**gazebo::physics::DARTMultiRayShape**
DART specific version of **MultiRayShape** (p. 664) . . . 328
**gazebo::physics::DARTPhysics**
DART physics engine . . . 330
**gazebo::physics::DARTPlaneShape**
An DART Plane shape . . . 336
**gazebo::physics::DARTRayShape**
Ray collision . . . 338
**gazebo::physics::DARTScrewJoint**
A screw joint . . . 339
**gazebo::physics::DARTSliderJoint**
A slider joint . . . 345
**gazebo::physics::DARTSphereShape**
A DART sphere shape . . . 350
**gazebo::physics::DARTTypes**
A set of functions for converting between the math types used by gazebo and dart . . . 351
**gazebo::physics::DARTUniversalJoint**
A universal joint . . . 352
**gazebo::rendering::DepthCamera**
Depth camera used to render depth data into an image buffer . . . 357
**gazebo::sensors::DepthCameraSensor** . . . 362
**gazebo::util::DiagnosticManager**
A diagnostic manager class . . . 365
**gazebo::util::DiagnosticTimer**
A timer designed for diagnostics . . . 368
**gazebo::rendering::DummyPageProvider**
Pretends to provide procedural page content to avoid page loading . . . 370
**gazebo::rendering::DynamicLines**
Class for drawing lines that can change . . . 371
**gazebo::rendering::DynamicRenderable**
Abstract base class providing mechanisms for dynamically growing hardware buffers . . . 375
**gazebo::physics::Entity**
**Base** (p. 159) class for all physics objects in Gazebo . . . 379
**gazebo::event::Event**
Base class for all events . . . 390
**gazebo::event::Events**
An **Event** (p. 390) class to get notifications for simulator events . . . 393
**gazebo::rendering::Events**
Base class for rendering events . . . 405
**gazebo::event::EventT< T >**
A class for event processing . . . 407
**gazebo::common::Exception**
Class for generating exceptions . . . 416
**gazebo::sensors::ForceTorqueSensor**
**Sensor** (p. 837) for measure force and torque on a joint . . . 418
**gazebo::rendering::FPSViewController**
First Person Shooter style view controller . . . 422
**google::protobuf::compiler::cpp::GazeboGenerator**
Google protobuf message generator for **gazebo::msgs** (p. 108) . . . 425
**gazebo::sensors::GpsSensor**
**GpsSensor** (p. 426) to provide position measurement . . . 426

**gazebo::rendering::GpuLaser**
GPU based laser distance sensor . . . 429
**gazebo::sensors::GpuRaySensor** . . . 438
**gazebo::rendering::Grid**
Displays a grid of cells, drawn with lines . . . 450
**gazebo::physics::Gripper**
A gripper abstraction . . . 453
**gazebo::rendering::GUIOverlay**
A class that creates a CEGUI overlay on a render window . . . 455
**gazebo::rendering::GzTerrainMatGen** . . . 459
**gazebo::rendering::Heightmap**
Rendering a terrain using heightmap information . . . 460
**gazebo::physics::HeightmapShape**
**HeightmapShape** (p. 465) collision shape builds a heightmap from an image . . . 465
**gazebo::physics::Hinge2Joint< T >**
A two axis hinge joint . . . 471
**gazebo::physics::HingeJoint< T >**
A single axis hinge joint . . . 472
**gazebo::common::Image**
Encapsulates an image . . . 474
**gazebo::sensors::ImuSensor**
An IMU sensor . . . 480
**gazebo::physics::Inertial**
A class for inertial information about a link . . . 483
**gazebo::common::InternalError**
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs . . . 493
**gazebo::transport::IOManager**
Manages boost::asio IO . . . 494
**gazebo::physics::Joint**
**Base** (p. 159) class for all joints . . . 496
**Joint_TEST** . . . 516
**gazebo::physics::JointController**
A class for manipulating **physics::Joint** (p. 496) . . . 521
**gazebo::physics::JointState**
Keeps track of state of a **physics::Joint** (p. 496) . . . 522
**gazebo::rendering::JointVisual**
Visualization for joints . . . 527
**gazebo::physics::JointWrench**
Wrench information from a joint . . . 529
**gazebo::common::KeyEvent**
Generic description of a keyboard event . . . 531
**gazebo::common::KeyFrame**
A key frame in an animation . . . 532
**gazebo::rendering::LaserVisual**
Visualization for laser data . . . 534
**gazebo::rendering::Light**
A light source . . . 535
**gazebo::physics::Link**
**Link** (p. 542) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body . . . 542
**gazebo::physics::LinkState**
Store state information of a **physics::Link** (p. 542) object . . . 563
**gazebo::util::LogPlay** . . . 570

**Logplay**
Open and playback log files that were recorded using LogRecord . . . 573
**gazebo::util::LogRecord**
Addtogroup gazebo_util . . . 573
**gazebo::physics::MapShape**
Creates box extrusions based on an image . . . 579
**gazebo::Master**
A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication . . . 583
**gazebo::common::Material**
Encapsulates description of a material . . . 585
**gazebo::math::Matrix3**
A 3x3 matrix class . . . 594
**gazebo::math::Matrix4**
A 3x3 matrix class . . . 599
**gazebo::common::Mesh**
A 3D mesh . . . 606
**gazebo::common::MeshCSG**
Creates CSG meshes . . . 613
**gazebo::common::MeshLoader**
Base class for loading meshes . . . 614
**gazebo::common::MeshManager**
Maintains and manages all meshes . . . 616
**gazebo::physics::MeshShape**
Triangle mesh collision shape . . . 621
**gazebo::physics::Model**
A model is a collection of links, joints, and plugins . . . 624
**gazebo::common::ModelDatabase**
Connects to model database, and has utility functions to find models . . . 638
**gazebo::ModelPlugin**
A plugin with access to **physics::Model** (p. 624) . . . 639
**gazebo::physics::ModelState**
Store state information of a **physics::Model** (p. 624) object . . . 641
**gazebo::common::MouseEvent**
Generic description of a mouse event . . . 649
**gazebo::rendering::MovableText**
Movable text . . . 652
**gazebo::msgs::MsgFactory**
A factory that generates protobuf message based on a string type . . . 658
**gazebo::sensors::MultiCameraSensor**
Multiple camera sensor . . . 660
**gazebo::physics::MultiRayShape**
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner . . . 664
**gazebo::transport::Node**
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics . . . 672
**gazebo::common::NodeAnimation**
Node animation . . . 679
**gazebo::common::NodeAssignment**
Vertex to node weighted assignement for skeleton animation visualization . . . 683
**gazebo::common::NodeTransform**
**NodeTransform** (p. 684) **Skeleton.hh** (p. 1367) common/common.hh . . . 684
**gazebo::sensors::Noise**
**Noise** (p. 689) models for sensor output signals . . . 689

**gazebo::common::NumericAnimation**
A numeric animation . . . 692
**gazebo::common::NumericKeyFrame**
A keyframe for a **NumericAnimation** (p. 692) . . . 693
**gazebo::util::OpenAL**
3D audio setup and playback . . . 695
**gazebo::util::OpenALSink**
**OpenAL** (p. 695) Listener . . . 697
**gazebo::util::OpenALSource**
**OpenAL** (p. 695) Source . . . 698
**gazebo::rendering::OrbitViewController**
Orbit view controller . . . 703
**gazebo::common::ParamT< T >** . . . 706
**gazebo::physics::PhysicsEngine**
**Base** (p. 159) class for a physics engine . . . 706
**gazebo::physics::PhysicsFactory**
The physics factory instantiates different physics engines . . . 720
**gazebo::common::PID**
Generic **PID** (p. 722) controller class . . . 722
**gazebo::math::Plane**
A plane and related functions . . . 726
**gazebo::physics::PlaneShape**
**Collision** (p. 220) for an infinite plane . . . 728
**gazebo::PluginT< T >**
A class which all plugins must inherit from . . . 732
**gazebo::math::Pose**
Encapsulates a position and rotation in three space . . . 734
**gazebo::common::PoseAnimation**
A pose animation . . . 743
**gazebo::common::PoseKeyFrame**
A keyframe for a **PoseAnimation** (p. 743) . . . 746
**gazebo::rendering::Projector**
Projects a material onto surface, light a light projector . . . 748
**gazebo::transport::Publication**
A publication for a topic . . . 750
**gazebo::transport::PublicationTransport**
Transport/transport.hh . . . 755
**gazebo::transport::Publisher**
A publisher of messages on a topic . . . 757
**QuadNode** . . . 760
**gazebo::math::Quaternion**
A quaternion class . . . 761
**gazebo::math::Rand**
Random number generator class . . . 775
**gazebo::transport::RawCallbackHelper**
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher . . . 776
**gazebo::sensors::RaySensor**
**Sensor** (p. 837) with one or more rays . . . 779
**gazebo::physics::RayShape**
**Base** (p. 159) class for Ray collision geometry . . . 786
**gazebo::rendering::RenderEngine**
Adaptor to Ogre3d . . . 791

**gazebo::sensors::RFIDSensor**
**Sensor** (p. 837) class for RFID type of sensor . . . 796
**gazebo::sensors::RFIDTag**
**RFIDTag** (p. 798) to interact with RFIDTagSensors . . . 798
**gazebo::rendering::RFIDTagVisual**
Visualization for RFID tags sensor . . . 801
**gazebo::rendering::RFIDVisual**
Visualization for RFID sensor . . . 802
**Road**
Used to render a strip of road . . . 804
**gazebo::physics::Road**
For building a **Road** (p. 804) from SDF . . . 804
**gazebo::rendering::Road2d** . . . 806
**gazebo::math::RotationSpline**
**Spline** (p. 993) for rotations . . . 806
**gazebo::rendering::RTShaderSystem**
Implements **Ogre** (p. 130)'s Run-Time Shader system . . . 810
**gazebo::rendering::Scene**
Representation of an entire scene graph . . . 814
**gazebo::physics::ScrewJoint< T >**
A screw joint, which has both prismatic and rotational DOFs . . . 832
**gazebo::rendering::SelectionObj**
Interactive selection object for models and links . . . 835
**gazebo::sensors::Sensor**
Base class for sensors . . . 837
**SensorFactor**
The sensor factory; the class is just for namespacing purposes . . . 848
**gazebo::sensors::SensorFactory** . . . 848
**gazebo::sensors::SensorManager**
Class to manage and update all sensors . . . 850
**gazebo::SensorPlugin**
A plugin with access to physics::Sensor . . . 853
**gazebo::Server** . . . 855
**gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg**
Keeping the CG shader for reference . . . 857
**gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL**
Utility class to help with generating shaders for GLSL . . . 859
**gazebo::physics::Shape**
**Base** (p. 159) class for all shapes . . . 861
**gazebo::physics::SimbodyBallJoint**
**SimbodyBallJoint** (p. 865) class models a ball joint in Simbody . . . 865
**gazebo::physics::SimbodyBoxShape**
Simbody box collision . . . 871
**gazebo::physics::SimbodyCollision**
Simbody collisions . . . 872
**gazebo::physics::SimbodyCylinderShape**
Cylinder collision . . . 875
**gazebo::physics::SimbodyHeightmapShape**
Height map collision . . . 877
**gazebo::physics::SimbodyHinge2Joint**
A two axis hinge joint . . . 879
**gazebo::physics::SimbodyHingeJoint**
A single axis hinge joint . . . 886

**gazebo::physics::SimbodyJoint**
**Base** (p. 159) class for all joints . . . 892
**gazebo::physics::SimbodyLink**
Simbody **Link** (p. 542) class . . . 900
**gazebo::physics::SimbodyMeshShape**
Triangle mesh collision . . . 910
**gazebo::physics::SimbodyModel**
A model is a collection of links, joints, and plugins . . . 912
**gazebo::physics::SimbodyMultiRayShape**
Simbody specific version of **MultiRayShape** (p. 664) . . . 913
**gazebo::physics::SimbodyPhysics**
Simbody physics engine . . . 915
**gazebo::physics::SimbodyPlaneShape**
Simbody collision for an infinite plane . . . 924
**gazebo::physics::SimbodyRayShape**
Ray shape for simbody . . . 926
**gazebo::physics::SimbodyScrewJoint**
A screw joint . . . 929
**gazebo::physics::SimbodySliderJoint**
A slider joint . . . 936
**gazebo::physics::SimbodySphereShape**
Simbody sphere collision . . . 942
**gazebo::physics::SimbodyUniversalJoint**
A simbody universal joint class . . . 944
**SingletonT< T >**
Singleton template class . . . 951
**gazebo::common::Skeleton**
A skeleton . . . 953
**gazebo::common::SkeletonAnimation**
**Skeleton** (p. 953) animation . . . 960
**gazebo::common::SkeletonNode**
A skeleton node . . . 964
**gazebo::physics::SliderJoint< T >**
A slider joint . . . 973
**gazebo::rendering::GzTerrainMatGen::SM2Profile**
Shader model 2 profile target . . . 975
**gazebo::sensors::SonarSensor**
**Sensor** (p. 837) with sonar cone . . . 977
**gazebo::rendering::SonarVisual**
Visualization for sonar data . . . 982
**Joint_TEST::SpawnJointOptions**
Class to hold parameters for spawning joints . . . 983
**gazebo::physics::SphereShape**
Sphere collision shape . . . 986
**gazebo::common::SphericalCoordinates**
Convert spherical coordinates for planetary surfaces . . . 988
**gazebo::math::Spline**
Splines . . . 993
**gazebo::physics::State**
**State** (p. 998) of an entity . . . 998
**gazebo::common::STLLoader**
Class used to load STL mesh files . . . 1002
**gazebo::common::SubMesh**
A child mesh . . . 1004

**gazebo::transport::SubscribeOptions**
Options for a subscription . . . 1014
**gazebo::transport::Subscriber**
A subscriber to a topic . . . 1016
**gazebo::transport::SubscriptionTransport**
Transport/transport.hh . . . 1018
**gazebo::physics::SurfaceParams**
**SurfaceParams** (p. 1020) defines various Surface contact parameters . . . 1020
**gazebo::common::SystemPaths**
Functions to handle getting system paths, keeps track of: . . . 1025
**gazebo::SystemPlugin**
A plugin loaded within the gzserver on startup . . . 1030
**gazebo::common::Time**
A **Time** (p. 1031) class, can be used to hold wall- or sim-time . . . 1031
**gazebo::common::Timer**
A timer class, used to time things in real world walltime . . . 1053
**gazebo::transport::TopicManager**
Manages topics and their subscriptions . . . 1055
**gazebo::physics::TrajectoryInfo** . . . 1061
**gazebo::rendering::TransmitterVisual**
Visualization for the wireless propagation data . . . 1062
**gazebo::physics::UniversalJoint< T >**
A universal joint . . . 1064
**gazebo::common::UpdateInfo**
Information for use in an update event . . . 1065
**gazebo::rendering::UserCamera**
A camera used for user visualization of a scene . . . 1066
**gazebo::math::Vector2d**
Generic double x, y vector . . . 1074
**gazebo::math::Vector2i**
Generic integer x, y vector . . . 1083
**gazebo::math::Vector3**
Generic vector containing 3 elements . . . 1091
**gazebo::math::Vector4**
Double Generic x, y, z, w vector . . . 1105
**gazebo::common::Video**
Handle video encoding and decoding using libavcodec . . . 1115
**gazebo::rendering::VideoVisual**
A visual element that displays a video as a texture . . . 1116
**gazebo::rendering::ViewController**
Base class for view controllers . . . 1118
**gazebo::rendering::Visual**
A renderable object . . . 1121
**gazebo::VisualPlugin**
A plugin loaded within the gzserver on startup . . . 1142
**gazebo::rendering::WindowManager**
Class to mangage render windows . . . 1144
**gazebo::rendering::WireBox**
Draws a wireframe box . . . 1146
**gazebo::sensors::WirelessReceiver**
**Sensor** (p. 837) class for receiving wireless signals . . . 1147
**gazebo::sensors::WirelessTransceiver**
**Sensor** (p. 837) class for receiving wireless signals . . . 1150

**gazebo::sensors::WirelessTransmitter**
Transmitter to send wireless signals . . . . . 1153
**gazebo::physics::World**
The world provides access to all other object within a simulated environment . . . . . 1157
**gazebo::WorldPlugin**
A plugin with access to **physics::World** (p. 1157) . . . . . 1169
**gazebo::physics::WorldState**
Store state information of a **physics::World** (p. 1157) object . . . . . 1170
**gazebo::rendering::WrenchVisual**
Visualization for sonar data . . . . . 1177

# Chapter 7

# File Index

## 7.1 File List

Here is a list of all files with brief descriptions:


**Actor.hh** . . . 1179
**Angle.hh** . . . 1180
**Animation.hh** . . . 1183
**ArrowVisual.hh** . . . 1184
**Assert.hh** . . . 1185
**AudioDecoder.hh** . . . 1187
**AxisVisual.hh** . . . 1188
**BallJoint.hh** . . . 1188
**Base.hh** . . . 1189
**Base64.hh** . . . 1190
**Box.hh** . . . 1192
**BoxShape.hh** . . . 1193
**BVHLoader.hh** . . . 1194
**CallbackHelper.hh** . . . 1196
**Camera.hh** . . . 1198
**CameraSensor.hh** . . . 1199
**CameraVisual.hh** . . . 1199
**cegui.h** . . . 1200
**ColladaLoader.hh** . . . 1201
**Collision.hh** . . . 1202
**CollisionState.hh** . . . 1203
**Color.hh** . . . 1203
**CommonIface.hh** . . . 1204
**CommonTypes.hh** . . . 1206
**COMVisual.hh** . . . 1208
**Connection.hh** . . . 1209
**ConnectionManager.hh** . . . 1211
**Console.hh** . . . 1213
**Contact.hh** . . . 1214
**ContactManager.hh** . . . 1216
**ContactSensor.hh** . . . 1217
**ContactVisual.hh** . . . 1217
**Conversions.hh** . . . 1218
**CylinderShape.hh** . . . 1219

**dart_inc.h** . . . 1220
**DARTBallJoint.hh** . . . 1221
**DARTBoxShape.hh** . . . 1222
**DARTCollision.hh** . . . 1222
**DARTCylinderShape.hh** . . . 1223
**DARTHeightmapShape.hh** . . . 1224
**DARTHinge2Joint.hh** . . . 1224
**DARTHingeJoint.hh** . . . 1225
**DARTJoint.hh** . . . 1226
**DARTLink.hh** . . . 1226
**DARTMeshShape.hh** . . . 1227
**DARTModel.hh** . . . 1227
**DARTMultiRayShape.hh** . . . 1228
**DARTPhysics.hh** . . . 1229
**DARTPlaneShape.hh** . . . 1229
**DARTRayShape.hh** . . . 1230
**DARTScrewJoint.hh** . . . 1230
**DARTSliderJoint.hh** . . . 1231
**DARTSphereShape.hh** . . . 1231
**DARTTypes.hh**
DART wrapper forward declarations and typedefs . . . 1232
**DARTUniversalJoint.hh** . . . 1233
**DepthCamera.hh** . . . 1234
**DepthCameraSensor.hh** . . . 1234
**Diagnostics.hh** . . . 1235
**DynamicLines.hh** . . . 1236
**DynamicRenderable.hh** . . . 1237
**Entity.hh** . . . 1237
**Event.hh** . . . 1238
**Events.hh** . . . 1239
**Exception.hh** . . . 1240
**ForceTorqueSensor.hh** . . . 1242
**FPSViewController.hh** . . . 1242
**gazebo.hh** . . . 1243
**gazebo_core.hh** . . . 1244
**GazeboGenerator.hh** . . . 1245
**GpsSensor.hh** . . . 1246
**GpuLaser.hh** . . . 1247
**GpuRaySensor.hh** . . . 1247
**Grid.hh** . . . 1248
**Gripper.hh** . . . 1249
**GUIOverlay.hh** . . . 1250
**Heightmap.hh** . . . 1250
**HeightmapShape.hh** . . . 1251
**Helpers.hh** . . . 1252
**Hinge2Joint.hh** . . . 1255
**HingeJoint.hh** . . . 1256
**Image.hh** . . . 1257
**ImuSensor.hh** . . . 1258
**Inertial.hh** . . . 1258
**IOManager.hh** . . . 1259
**Joint.hh** . . . 1261
**Joint_TEST.hh** . . . 1262
**JointController.hh** . . . 1263

**JointState.hh** . . . 1264
**JointVisual.hh** . . . 1265
**JointWrench.hh** . . . 1265
**KeyEvent.hh** . . . 1266
**KeyFrame.hh** . . . 1267
**LaserVisual.hh** . . . 1268
**Light.hh** . . . 1269
**Link.hh** . . . 1270
**LinkState.hh** . . . 1271
**LogPlay.hh** . . . 1273
**LogRecord.hh** . . . 1273
**mainpage.html** . . . 1274
**MapShape.hh** . . . 1274
**Master.hh** . . . 1275
**common/Material.hh** . . . 1276
**rendering/Material.hh** . . . 1278
**MathTypes.hh**
Forward declarations for the math classes . . . 1278
**Matrix3.hh** . . . 1278
**Matrix4.hh** . . . 1279
**Mesh.hh** . . . 1280
**MeshCSG.hh** . . . 1282
**MeshLoader.hh** . . . 1284
**MeshManager.hh** . . . 1285
**MeshShape.hh** . . . 1286
**Model.hh** . . . 1287
**ModelDatabase.hh** . . . 1289
**ModelState.hh** . . . 1290
**MouseEvent.hh** . . . 1291
**MovableText.hh** . . . 1293
**MsgFactory.hh** . . . 1293
**msgs.hh** . . . 1294
**MultiCameraSensor.hh** . . . 1297
**MultiRayShape.hh** . . . 1298
**Node.hh** . . . 1299
**Noise.hh** . . . 1300
**ogre_gazebo.h** . . . 1301
**OpenAL.hh** . . . 1302
**OrbitViewController.hh** . . . 1303
**PhysicsEngine.hh** . . . 1303
**PhysicsFactory.hh** . . . 1304
**PhysicsIface.hh** . . . 1306
**PhysicsTypes.hh**
Default namespace for gazebo . . . 1308
**PID.hh** . . . 1311
**Plane.hh** . . . 1312
**PlaneShape.hh** . . . 1312
**Plugin.hh** . . . 1313
**Pose.hh** . . . 1317
**Projector.hh** . . . 1318
**Publication.hh** . . . 1318
**PublicationTransport.hh** . . . 1321
**Publisher.hh** . . . 1323
**Quaternion.hh** . . . 1325

**Rand.hh** . . . 1326
**RaySensor.hh** . . . 1327
**RayShape.hh** . . . 1328
**RenderEngine.hh** . . . 1329
**RenderEvents.hh** . . . 1330
**RenderingIface.hh** . . . 1330
**RenderTypes.hh** . . . 1332
**RFIDSensor.hh** . . . 1334
**RFIDTag.hh** . . . 1335
**RFIDTagVisual.hh** . . . 1335
**RFIDVisual.hh** . . . 1336
**Road.hh** . . . 1337
**Road2d.hh** . . . 1338
**RotationSpline.hh** . . . 1338
**RTShaderSystem.hh** . . . 1340
**Scene.hh** . . . 1340
**ScrewJoint.hh** . . . 1342
**SelectionObj.hh** . . . 1343
**Sensor.hh** . . . 1343
**SensorFactory.hh** . . . 1344
**SensorManager.hh** . . . 1345
**SensorsIface.hh** . . . 1346
**SensorTypes.hh**
Forward declarations and typedefs for sensors . . . 1348
**Server.hh** . . . 1350
**Shape.hh** . . . 1351
**simbody_inc.h** . . . 1352
**SimbodyBallJoint.hh** . . . 1353
**SimbodyBoxShape.hh** . . . 1354
**SimbodyCollision.hh** . . . 1354
**SimbodyCylinderShape.hh** . . . 1355
**SimbodyHeightmapShape.hh** . . . 1355
**SimbodyHinge2Joint.hh** . . . 1356
**SimbodyHingeJoint.hh** . . . 1357
**SimbodyJoint.hh** . . . 1357
**SimbodyLink.hh** . . . 1358
**SimbodyMeshShape.hh** . . . 1359
**SimbodyModel.hh** . . . 1359
**SimbodyMultiRayShape.hh** . . . 1360
**SimbodyPhysics.hh** . . . 1360
**SimbodyPlaneShape.hh** . . . 1361
**SimbodyRayShape.hh** . . . 1362
**SimbodyScrewJoint.hh** . . . 1363
**SimbodySliderJoint.hh** . . . 1363
**SimbodySphereShape.hh** . . . 1364
**SimbodyTypes.hh**
Simbody wrapper forward declarations and typedefs . . . 1364
**SimbodyUniversalJoint.hh** . . . 1366
**SingletonT.hh** . . . 1366
**Skeleton.hh** . . . 1367
**SkeletonAnimation.hh** . . . 1369
**SliderJoint.hh** . . . 1370
**SonarSensor.hh** . . . 1371
**SonarVisual.hh** . . . 1372

**SphereShape.hh** . . . 1373
**SphericalCoordinates.hh** . . . 1374
**Spline.hh** . . . 1375
**State.hh** . . . 1377
**STLLoader.hh** . . . 1378
**SubscribeOptions.hh** . . . 1379
**Subscriber.hh** . . . 1381
**SubscriptionTransport.hh** . . . 1383
**SurfaceParams.hh** . . . 1385
**SystemPaths.hh** . . . 1386
**Time.hh** . . . 1387
**Timer.hh** . . . 1388
**TopicManager.hh** . . . 1389
**TransmitterVisual.hh** . . . 1391
**TransportIface.hh** . . . 1392
**TransportTypes.hh**
Forward declarations for transport . . . 1394
**UniversalJoint.hh** . . . 1395
**UpdateInfo.hh** . . . 1396
**UserCamera.hh** . . . 1397
**UtilTypes.hh** . . . 1398
**Vector2d.hh** . . . 1399
**Vector2i.hh** . . . 1400
**Vector3.hh** . . . 1401
**Vector4.hh** . . . 1401
**Video.hh** . . . 1403
**VideoVisual.hh** . . . 1404
**ViewController.hh** . . . 1405
**Visual.hh** . . . 1406
**WindowManager.hh** . . . 1407
**WireBox.hh** . . . 1408
**WirelessReceiver.hh** . . . 1409
**WirelessTransceiver.hh** . . . 1409
**WirelessTransmitter.hh** . . . 1410
**World.hh** . . . 1411
**WorldState.hh** . . . 1412
**WrenchVisual.hh** . . . 1414

# Chapter 8

# Module Documentation

## 8.1 Common

### Files

- file **CommonTypes.hh**

### Namespaces

- namespace **gazebo::common**

  *Common namespace.*

### Classes

- class **gazebo::common::Animation**

  *Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.*
- class **gazebo::common::AssertionInternalError**

  *Class for generating Exceptions which come from gazebo assertions.*
- class **gazebo::common::AudioDecoder**

  *An audio decoder based on FFMPEG.*
- class **gazebo::common::BVHLoader**

  *Handles loading BVH animation files.*
- class **gazebo::common::ColladaLoader**

  *Class used to load Collada mesh files.*
- class **gazebo::common::Color**

  *Defines a color.*
- class **gazebo::common::Console**

  *Message, error, warning functionality.*
- class **gazebo::common::Exception**

  *Class for generating exceptions.*
- class **gazebo::common::Image**

  *Encapsulates an image.*
- class **gazebo::common::InternalError**

*Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.*

- class **gazebo::common::KeyEvent**

  *Generic description of a keyboard event.*
- class **gazebo::common::KeyFrame**

  *A key frame in an animation.*
- class **gazebo::common::Material**

  *Encapsulates description of a material.*
- class **gazebo::common::Mesh**

  *A 3D mesh.*
- class **gazebo::common::MeshCSG**

  *Creates CSG meshes.*
- class **gazebo::common::MeshLoader**

  *Base class for loading meshes.*
- class **gazebo::common::MeshManager**

  *Maintains and manages all meshes.*
- class **gazebo::common::ModelDatabase**

  *Connects to model database, and has utility functions to find models.*
- class **gazebo::ModelPlugin**

  *A plugin with access to **physics::Model** (p. 624).*
- class **gazebo::common::MouseEvent**

  *Generic description of a mouse event.*
- class **gazebo::common::NodeAnimation**

  *Node animation.*
- struct **gazebo::common::NodeAssignment**

  *Vertex to node weighted assignement for skeleton animation visualization.*
- class **gazebo::common::NodeTransform**

  ***NodeTransform** (p. 684) **Skeleton.hh** (p. 1367) common/common.hh*
- class **gazebo::common::NumericAnimation**

  *A numeric animation.*
- class **gazebo::common::NumericKeyFrame**

  *A keyframe for a **NumericAnimation** (p. 692).*
- class **gazebo::common::PID**

  *Generic **PID** (p. 722) controller class.*
- class **gazebo::PluginT< T >**

  *A class which all plugins must inherit from.*
- class **gazebo::common::PoseAnimation**

  *A pose animation.*
- class **gazebo::common::PoseKeyFrame**

  *A keyframe for a **PoseAnimation** (p. 743).*
- class **gazebo::SensorPlugin**

  *A plugin with access to physics::Sensor.*
- class **SingletonT< T >**

  *Singleton template class.*
- class **gazebo::common::Skeleton**

  *A skeleton.*
- class **gazebo::common::SkeletonAnimation**

  ***Skeleton** (p. 953) animation.*

- class **gazebo::common::SkeletonNode**
  *A skeleton node.*
- class **gazebo::common::SphericalCoordinates**
  *Convert spherical coordinates for planetary surfaces.*
- class **gazebo::common::STLLoader**
  *Class used to load STL mesh files.*
- class **gazebo::common::SubMesh**
  *A child mesh.*
- class **gazebo::common::SystemPaths**
  *Functions to handle getting system paths, keeps track of:*
- class **gazebo::SystemPlugin**
  *A plugin loaded within the gzserver on startup.*
- class **gazebo::common::Time**
  *A **Time** (p. 1031) class, can be used to hold wall- or sim-time.*
- class **gazebo::common::Timer**
  *A timer class, used to time things in real world walltime.*
- class **gazebo::common::Video**
  *Handle video encoding and decoding using libavcodec.*
- class **gazebo::VisualPlugin**
  *A plugin loaded within the gzserver on startup.*
- class **gazebo::WorldPlugin**
  *A plugin with access to **physics::World** (p. 1157).*

## Macros

- #define **gzclr_end** "\033[0m"
  *End marker.*
- #define **gzclr_start**(clr) "\033[1;33m"
  *Start marker.*
- #define **gzdbg** (**gazebo::common::Console::Instance**()->ColorMsg("Dbg", 36))
  *Output a debug message.*
- #define **gzerr**
  *Output an error message.*
- #define **gzlog** (**gazebo::common::Console::Instance**()->Log())
  *Output a message to a log file.*
- #define **gzmsg** (**gazebo::common::Console::Instance**()->ColorMsg("Msg", 32))
  *Output a message.*
- #define **gzthrow**(msg)
  *This macro logs an error to the throw stream and throws an exception that contains the file name and line number.*
- #define **gzwarn**
  *Output a warning message.*

## Enumerations

- enum **gazebo::PluginType** {
  **gazebo::WORLD_PLUGIN**, **gazebo::MODEL_PLUGIN**, **gazebo::SENSOR_PLUGIN**, **gazebo::SYSTEM_PLUGIN**,
  **gazebo::VISUAL_PLUGIN** }
  *Used to specify the type of plugin.*

## Functions

- **gazebo::common::Console::NullStream::NullStream** ()

  *constructor*
- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)

  *add path sufix to **common::SystemPaths*** (p. *1025)*
- std::ostream & **gazebo::common::Console::ColorErr** (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)

  *Use this to output an error to the terminal.*
- std::ostream & **gazebo::common::Console::ColorMsg** (const std::string &_lbl, int _color)

  *Use this to output a colored message to the terminal.*
- void **gazebo::common::ModelDatabase::DownloadDependencies** (const std::string &_path)

  *Download all dependencies for a give model path.*
- std::string **gazebo::common::find_file** (const std::string &_file)

  *search for file in **common::SystemPaths*** (p. *1025)*
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)

  *search for file in **common::SystemPaths*** (p. *1025)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths*** (p. *1025)*
- void **gazebo::common::ModelDatabase::Fini** ()

  *Finalize the model database.*
- template<typename T >
  std::string **gazebo::common::get_sha1** (const T &_buffer)

  *Compute the SHA1 hash of an array of bytes.*
- std::string **gazebo::common::ModelDatabase::GetDBConfig** (const std::string &_uri)

  *Return the database.config file as a string.*
- std::string **gazebo::common::ModelDatabase::GetModelConfig** (const std::string &_uri)

  *Return the model.config file as a string.*
- std::string **gazebo::common::ModelDatabase::GetModelFile** (const std::string &_uri)

  *Get a model's SDF file based on a URI.*
- std::string **gazebo::common::ModelDatabase::GetModelName** (const std::string &_uri)

  *Get the name of a model based on a URI.*
- std::string **gazebo::common::ModelDatabase::GetModelPath** (const std::string &_uri, bool _forceDownload=false)

  *Get the local path to a model.*
- std::map< std::string,
  std::string > **gazebo::common::ModelDatabase::GetModels** ()

  *Returns the dictionary of all the model names.*
- void **gazebo::common::ModelDatabase::GetModels** (boost::function< void(const std::map< std::string, std::string > &)> _func)

  *Get the dictionary of all model names via a callback.*
- bool **gazebo::common::Console::GetQuiet** () const

  *Get whether quiet output is set.*
- std::string **gazebo::common::ModelDatabase::GetURI** ()

  *Returns the the global model database URI.*
- bool **gazebo::common::ModelDatabase::HasModel** (const std::string &_modelName)

  *Returns true if the model exists on the database.*
- void **gazebo::common::Console::Init** (const std::string &_logFilename)

*Load the message parameters.*

- bool **gazebo::common::Console::IsInitialized** () const

  *Return true if Init has been called.*

- void **gazebo::common::load** ()

  *Load the common library.*

- std::ofstream & **gazebo::common::Console::Log** ()

  *Use this to output a colored message to the terminal.*

- void **gazebo::common::Console::SetQuiet** (bool _q)

  *Set quiet output.*

- void **gazebo::common::ModelDatabase::Start** (bool _fetchImmediately=false)

  *Start the model database.*

### Variables

- static std::string **gazebo::common::PixelFormatNames** [ ]

  *String names for the pixel formats.*

## 8.1.1 Detailed Description

## 8.1.2 Macro Definition Documentation

#### 8.1.2.1 #define gzclr_end "\033[0m"

End marker.

#### 8.1.2.2 #define gzclr_start( *clr* ) "\033[1;33m"

Start marker.

#### 8.1.2.3 #define gzdbg (gazebo::common::Console::Instance()->ColorMsg("Dbg", 36))

Output a debug message.

Referenced by Joint_TEST::SetUp().

#### 8.1.2.4 #define gzerr

**Value:**

```
(gazebo::common::Console::Instance()->ColorErr("Error", \
          __FILE__, __LINE__, 31))
```

Output an error message.

Referenced by gazebo::transport::Connection::AsyncRead(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::DARTBallJoint::SetForceImpl(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

#### 8.1.2.5 #define gzlog (gazebo::common::Console::Instance()->Log())

Output a message to a log file.

#### 8.1.2.6 #define gzmsg (gazebo::common::Console::Instance()->ColorMsg("Msg", 32))

Output a message.

#### 8.1.2.7 #define gzthrow( *msg* )

**Value:**

```
{std::ostringstream throwStream;\
      throwStream << msg << std::endl << std::flush;\
      throw gazebo::common::Exception(__FILE__, __LINE__, throwStream.str()); }
```

This macro logs an error to the throw stream and throws an exception that contains the file name and line number.

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), and gazebo::transport::SubscribeOptions::Init().

#### 8.1.2.8 #define gzwarn

**Value:**

```
(gazebo::common::Console::Instance()->ColorErr("Warning", \
          __FILE__, __LINE__, 33))
```

Output a warning message.

Referenced by gazebo::physics::ScrewJoint< SimbodyJoint >::Load(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), gazebo::physics::DARTBoxShape::SetSize(), and Joint_TEST::SpawnJoint().

### 8.1.3 Enumeration Type Documentation

#### 8.1.3.1 enum gazebo::PluginType

Used to specify the type of plugin.

**Enumerator**

***WORLD_PLUGIN*** A World plugin.

***MODEL_PLUGIN*** A Model plugin.

***SENSOR_PLUGIN*** A Sensor plugin.

***SYSTEM_PLUGIN*** A System plugin.

***VISUAL_PLUGIN*** A Visual plugin.

### 8.1.4 Function Documentation

#### 8.1.4.1 gazebo::common::Console::NullStream::NullStream ( ) [inline]

constructor

#### 8.1.4.2 void gazebo::common::add_search_path_suffix ( const std::string & _suffix )

add path sufix to **common::SystemPaths** (p. 1025)

**Parameters**

| | | |
|---|---|---|
| in | *_suffix* | The suffix to add. |

#### 8.1.4.3 std::ostream& gazebo::common::Console::ColorErr ( const std::string & _lbl, const std::string & _file, unsigned int _line, int _color )

Use this to output an error to the terminal.

**Parameters**

| | | |
|---|---|---|
| in | *_lbl* | Text label |
| in | *_file* | File containing the error |
| in | *_line* | Line containing the error |
| in | *_color* | **Color** (p. 233) to make the label |

**Returns**

Reference to an output stream

#### 8.1.4.4 std::ostream& gazebo::common::Console::ColorMsg ( const std::string & _lbl, int _color )

Use this to output a colored message to the terminal.

**Parameters**

| | | |
|---|---|---|
| in | *_lbl* | Text label |
| in | *_color* | **Color** (p. 233) to make the label |

**Returns**

Reference to an output stream

#### 8.1.4.5 void gazebo::common::ModelDatabase::DownloadDependencies ( const std::string & _path )

Download all dependencies for a give model path.

Look's in the model's manifest file (_path/model.config) for all models listed in the <depend> block, and downloads the models if necessary.

**Parameters**

| | | |
|---|---|---|
| in | *_path* | Path to a model. |

#### 8.1.4.6 std::string gazebo::common::find_file ( const std::string & *_file* )

search for file in **common::SystemPaths** (p. 1025)

**Parameters**

| | | |
|---|---|---|
| in | *_file* | Name of the file to find. |

**Returns**

The path containing the file.

#### 8.1.4.7 std::string gazebo::common::find_file ( const std::string & *_file,* bool *_searchLocalPath* )

search for file in **common::SystemPaths** (p. 1025)

**Parameters**

| | | |
|---|---|---|
| in | *_file* | Name of the file to find. |
| in | *_searchLocal-Path* | True to search in the current working directory. |

**Returns**

The path containing the file.

#### 8.1.4.8 std::string gazebo::common::find_file_path ( const std::string & *_file* )

search for a file in **common::SystemPaths** (p. 1025)

**Parameters**

| | | |
|---|---|---|
| in | *_file* | the file name to look for. |

**Returns**

The path containing the file.

#### 8.1.4.9 void gazebo::common::ModelDatabase::Fini ( )

Finalize the model database.

#### 8.1.4.10 template<typename T > std::string gazebo::common::get_sha1 ( const T & *_buffer* )

Compute the SHA1 hash of an array of bytes.

**Parameters**

| | | |
|---|---|---|
| in | *_buffer* | Input sequence. The permitted data types for this function are std::string and any STL container. |

**Returns**

The string representation (40 character) of the SHA1 hash.

References NULL.

#### 8.1.4.11 std::string gazebo::common::ModelDatabase::GetDBConfig ( const std::string & *_uri* )

Return the database.config file as a string.

**Returns**

The database config file from the model database.

#### 8.1.4.12 std::string gazebo::common::ModelDatabase::GetModelConfig ( const std::string & *_uri* )

Return the model.config file as a string.

**Returns**

The model config file from the model database.

#### 8.1.4.13 std::string gazebo::common::ModelDatabase::GetModelFile ( const std::string & *_uri* )

Get a model's SDF file based on a URI.

Get a model file based on a URI. If the model is on a remote server, then the model fetched and installed locally.

**Parameters**

| | | |
|---|---|---|
| in | *_uri* | The URI of the model |

**Returns**

The full path and filename to the SDF file

#### 8.1.4.14 std::string gazebo::common::ModelDatabase::GetModelName ( const std::string & *_uri* )

Get the name of a model based on a URI.

The URI must be fully qualified: `http://gazebosim.org/gazebo_models/ground_plane` or model-://gazebo_models

**Parameters**

| | | |
|---|---|---|
| in | *_uri* | the model uri |

**Returns**

the model's name.

#### 8.1.4.15 std::string gazebo::common::ModelDatabase::GetModelPath ( const std::string & _uri, bool _forceDownload = false )

Get the local path to a model.

Get the path to a model based on a URI. If the model is on a remote server, then the model fetched and installed locally.

**Parameters**

| | | |
|---|---|---|
| in | *_uri* | the model uri |
| in | *_forceDownload* | True to skip searching local paths. |

**Returns**

path to a model directory

#### 8.1.4.16 std::map<std::string, std::string> gazebo::common::ModelDatabase::GetModels ( )

Returns the dictionary of all the model names.

This is a blocking call. Which means it will wait for the **ModelDatabase** (p. 638) to download the model list.

**Returns**

a map of model names, indexed by their full URI.

#### 8.1.4.17 void gazebo::common::ModelDatabase::GetModels ( boost::function< void(const std::map< std::string, std::string > &)> _func )

Get the dictionary of all model names via a callback.

This is the non-blocking version of **ModelDatabase::GetModels** (p. 42)

**Parameters**

| | | |
|---|---|---|
| in | *_func* | Callback function that receives the list of models. |

#### 8.1.4.18 bool gazebo::common::Console::GetQuiet ( ) const

Get whether quiet output is set.

**Returns**

True to if quiet output is set

#### 8.1.4.19 std::string gazebo::common::ModelDatabase::GetURI ( )

Returns the the global model database URI.

**Returns**

the URI.

#### 8.1.4.20 bool gazebo::common::ModelDatabase::HasModel ( const std::string & _modelName )

Returns true if the model exists on the database.

**Parameters**

| | | |
|---|---|---|
| in | *_modelName* | URI of the model (eg: model://my_model_name). |

**Returns**

True if the model was found.

#### 8.1.4.21 void gazebo::common::Console::Init ( const std::string & _logFilename )

Load the message parameters.

#### 8.1.4.22 bool gazebo::common::Console::IsInitialized ( ) const

Return true if Init has been called.

**Returns**

True is initialized.

#### 8.1.4.23 void gazebo::common::load ( )

Load the common library.

#### 8.1.4.24 std::ofstream& gazebo::common::Console::Log ( )

Use this to output a colored message to the terminal.

**Parameters**

| | | |
|---|---|---|
| in | *_lbl* | Text label |

**Returns**

Reference to an output stream

#### 8.1.4.25 void gazebo::common::Console::SetQuiet ( bool _q )

Set quiet output.

**Parameters**

| | | |
|---|---|---|
| in | *q* | True to prevent warning |

#### 8.1.4.26 void gazebo::common::ModelDatabase::Start ( bool *_fetchImmediately* = false )

Start the model database.

**Parameters**

| | | |
|---|---|---|
| in | *_fetchImmediately* | True to fetch the models without waiting. |

### 8.1.5 Variable Documentation

#### 8.1.5.1 std::string gazebo::common::PixelFormatNames[] [static]

**Initial value:**

```
=
    {
      "UNKNOWN_PIXEL_FORMAT",
      "L_INT8",
      "L_INT16",
      "RGB_INT8",
      "RGBA_INT8",
      "BGRA_INT8",
      "RGB_INT16",
      "RGB_INT32",
      "BGR_INT8",
      "BGR_INT16",
      "BGR_INT32",
      "R_FLOAT16",
      "RGB_FLOAT16",
      "R_FLOAT32",
      "RGB_FLOAT32",
      "BAYER_RGGB8",
      "BAYER_RGGR8",
      "BAYER_GBRG8",
      "BAYER_GRBG8"
    }
```

String names for the pixel formats.

**See Also**

**Image::PixelFormat** (p. 476).

## 8.2 Events

### Namespaces

- namespace **gazebo::event**

  ***Event*** *(p. 390) namespace.*

### Classes

- class **gazebo::event::Connection**

  *A class that encapsulates a connection.*
- class **gazebo::event::Event**

  *Base class for all events.*
- class **gazebo::event::Events**

  *An* ***Event*** *(p. 390) class to get notifications for simulator events.*
- class **gazebo::event::EventT< T >**

  *A class for event processing.*

### Functions

- virtual **gazebo::event::EventT< T >::~EventT** ()

  *Destructor.*
- ConnectionPtr **gazebo::event::EventT< T >::Connect** (const boost::function< T > &_subscriber)

  *Connect a callback to this event.*
- unsigned int **gazebo::event::EventT< T >::ConnectionCount** () const

  *Get the number of connections.*
- virtual void **gazebo::event::EventT< T >::Disconnect** (ConnectionPtr _c)

  *Disconnect a callback to this event.*
- virtual void **gazebo::event::EventT< T >::Disconnect** (int _id)

  *Disconnect a callback to this event.*

### 8.2.1 Detailed Description

### 8.2.2 Function Documentation

#### 8.2.2.1 template<typename T > gazebo::event::EventT< T >::~EventT ( ) [virtual]

Destructor.

Destructor. Deletes all the associated connections.

#### 8.2.2.2 template<typename T> ConnectionPtr gazebo::event::EventT< T >::Connect ( const boost::function< T > & *_subscriber* )

Connect a callback to this event.

Adds a connection.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Pointer to a callback function |

**Returns**

A **Connection** (p. 246) object, which will automatically call Disconnect when it goes out of scope

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to connect |

Referenced by gazebo::event::Events::ConnectAddEntity(), gazebo::event::Events::ConnectCreateEntity(), gazebo::rendering::Events::ConnectCreateScene(), gazebo::event::Events::ConnectDeleteEntity(), gazebo::event::Events::ConnectDiagTimerStart(), gazebo::event::Events::ConnectDiagTimerStop(), gazebo::physics::Link::ConnectEnabled(), gazebo::physics::Joint::ConnectJointUpdate(), gazebo::rendering::DepthCamera::ConnectNewDepthFrame(), gazebo::rendering::Camera::ConnectNewImageFrame(), gazebo::rendering::GpuLaser::ConnectNewLaserFrame(), gazebo::physics::MultiRayShape::ConnectNewLaserScans(), gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud(), gazebo::event::Events::ConnectPause(), gazebo::event::Events::ConnectPostRender(), gazebo::event::Events::ConnectPreRender(), gazebo::rendering::Events::ConnectRemoveScene(), gazebo::event::Events::ConnectRender(), gazebo::event::Events::ConnectSetSelectedEntity(), gazebo::event::Events::ConnectSigInt(), gazebo::event::Events::ConnectStep(), gazebo::event::Events::ConnectStop(), gazebo::transport::Connection::ConnectToShutdown(), gazebo::sensors::ForceTorqueSensor::ConnectUpdate(), gazebo::sensors::SonarSensor::ConnectUpdate(), gazebo::sensors::Sensor::ConnectUpdated(), gazebo::event::Events::ConnectWorldCreated(), gazebo::event::Events::ConnectWorldUpdateBegin(), and gazebo::event::Events::ConnectWorldUpdateEnd().

#### 8.2.2.3 template<typename T > unsigned int gazebo::event::EventT< T >::ConnectionCount ( ) const

Get the number of connections.

**Returns**

Number of connection to this **Event** (p. 390).

#### 8.2.2.4 template<typename T > void gazebo::event::EventT< T >::Disconnect ( ConnectionPtr _c ) [virtual]

Disconnect a callback to this event.

Removes a connection.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The connection to disconnect |
| in | *_c* | the connection |

Implements **gazebo::event::Event** (p. 392).

References NULL.

Referenced by gazebo::event::Events::DisconnectAddEntity(), gazebo::event::Events::DisconnectCreateEntity(), gazebo::rendering::Events::DisconnectCreateScene(), gazebo::event::Events::DisconnectDeleteEntity(), gazebo::event::Events::DisconnectDiagTimerStart(), gazebo::event::Events::DisconnectDiagTimerStop(), gazebo::physics::Link::DisconnectEnabled(), gazebo::physics::Joint::DisconnectJointUpdate(), gazebo::rendering::DepthCamera::DisconnectNewDepthFrame(), gazebo::rendering::Camera::DisconnectNewImageFrame(), gazebo::rendering::-

GpuLaser::DisconnectNewLaserFrame(), gazebo::physics::MultiRayShape::DisconnectNewLaserScans(), gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud(), gazebo::event::Events::DisconnectPause(), gazebo::event::Events::DisconnectPostRender(), gazebo::event::Events::DisconnectPreRender(), gazebo::rendering::Events::DisconnectRemoveScene(), gazebo::event::Events::DisconnectRender(), gazebo::event::Events::DisconnectSetSelectedEntity(), gazebo::transport::Connection::DisconnectShutdown(), gazebo::event::Events::DisconnectSigInt(), gazebo::event::Events::DisconnectStep(), gazebo::event::Events::DisconnectStop(), gazebo::sensors::ForceTorqueSensor::DisconnectUpdate(), gazebo::sensors::SonarSensor::DisconnectUpdate(), gazebo::sensors::Sensor::DisconnectUpdated(), gazebo::event::Events::DisconnectWorldCreated(), and gazebo::event::Events::DisconnectWorldUpdateEnd().

#### 8.2.2.5 template<typename T > void gazebo::event::EventT< T >::Disconnect ( int _id ) [virtual]

Disconnect a callback to this event.

Removes a connection.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | The id of the connection to disconnect |
| in | *_id* | the connection index |

Implements **gazebo::event::Event** (p. 392).

## 8.3 Math

A set of classes that encapsulate math related properties and functions.

### Files

- file **MathTypes.hh**

  *Forward declarations for the math classes.*

### Namespaces

- namespace **gazebo::math**

  *Math namespace.*

### Classes

- class **gazebo::math::Angle**

  *An angle and related functions.*
- class **gazebo::math::Box**

  *Mathematical representation of a box and related functions.*
- class **gazebo::math::Matrix3**

  *A 3x3 matrix class.*
- class **gazebo::math::Matrix4**

  *A 3x3 matrix class.*
- class **gazebo::math::Plane**

  *A plane and related functions.*
- class **gazebo::math::Pose**

  *Encapsulates a position and rotation in three space.*
- class **gazebo::math::Quaternion**

  *A quaternion class.*
- class **gazebo::math::Rand**

  *Random number generator class.*
- class **gazebo::math::RotationSpline**

  ***Spline*** *(p. 993) for rotations.*
- class **gazebo::math::Spline**

  *Splines.*
- class **gazebo::math::Vector2d**

  *Generic double x, y vector.*
- class **gazebo::math::Vector2i**

  *Generic integer x, y vector.*
- class **gazebo::math::Vector3**

  *The* ***Vector3*** *(p. 1091) class represents the generic vector containing 3 elements.*
- class **gazebo::math::Vector4**

  *double Generic x, y, z, w vector*

## Functions

- template<typename T >
  T **gazebo::math::clamp** (T _v, T _min, T _max)

  *Simple clamping function.*
- template<typename T >
  bool **gazebo::math::equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

  *check if two values are equal, within a tolerance*
- float **gazebo::math::fixnan** (float _v)

  *Fix a nan value.*
- double **gazebo::math::fixnan** (double _v)

  *Fix a nan value.*
- bool **gazebo::math::isnan** (float _v)

  *check if a float is NaN*
- bool **gazebo::math::isnan** (double _v)

  *check if a double is NaN*
- bool **gazebo::math::isPowerOfTwo** (unsigned int _x)

  *is this a power of 2?*
- template<typename T >
  T **gazebo::math::max** (const std::vector< T > &_values)

  *get the maximum value of vector of values*
- template<typename T >
  T **gazebo::math::mean** (const std::vector< T > &_values)

  *get mean of vector of values*
- template<typename T >
  T **gazebo::math::min** (const std::vector< T > &_values)

  *get the minimum value of vector of values*
- double **gazebo::math::parseFloat** (const std::string &_input)

  *parse string into float*
- int **gazebo::math::parseInt** (const std::string &_input)

  *parse string into an integer*
- template<typename T >
  T **gazebo::math::precision** (const T &_a, const unsigned int &_precision)

  *get value at a specified precision*
- template<typename T >
  T **gazebo::math::variance** (const std::vector< T > &_values)

  *get variance of vector of values*

## Variables

- static const double **gazebo::math::NAN_D** = std::numeric_limits<double>::quiet_NaN()

  *Returns the representation of a quiet not a number (NAN)*
- static const int **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()

  *Returns the representation of a quiet not a number (NAN)*

### 8.3.1 Detailed Description

A set of classes that encapsulate math related properties and functions.

### 8.3.2 Function Documentation

#### 8.3.2.1 template<typename T > T gazebo::math::clamp ( T _v, T _min, T _max ) [inline]

Simple clamping function.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | value |
| in | *_min* | minimum |
| in | *_max* | maximum |

References gazebo::math::max(), and gazebo::math::min().

#### 8.3.2.2 template<typename T > bool gazebo::math::equal ( const T & _a, const T & _b, const T & _epsilon = 1e-6 ) [inline]

check if two values are equal, within a tolerance

**Parameters**

| | | |
|---|---|---|
| in | *_a* | the first value |
| in | *_b* | the second value |
| in | *_epsilon* | the tolerance |

Referenced by gazebo::math::Quaternion::Correct(), gazebo::math::Quaternion::GetInverse(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

#### 8.3.2.3 float gazebo::math::fixnan ( float _v ) [inline]

Fix a nan value.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | Value to correct. |

**Returns**

0 if _v is NaN, _v otherwise.

References gazebo::math::isnan().

#### 8.3.2.4 double gazebo::math::fixnan ( double _v ) [inline]

Fix a nan value.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | Value to correct. |

**Returns**

0 if _v is NaN, _v otherwise.

References gazebo::math::isnan().

#### 8.3.2.5 bool gazebo::math::isnan ( float _v ) [inline]

check if a float is NaN

**Parameters**

| | | |
|---|---|---|
| in | _v | the value |

**Returns**

true if _v is not a number, false otherwise

Referenced by gazebo::math::fixnan(), and gazebo::math::isnan().

#### 8.3.2.6 bool gazebo::math::isnan ( double _v ) [inline]

check if a double is NaN

**Parameters**

| | | |
|---|---|---|
| in | _v | the value |

**Returns**

true if _v is not a number, false otherwise

References gazebo::math::isnan().

#### 8.3.2.7 bool gazebo::math::isPowerOfTwo ( unsigned int _x ) [inline]

is this a power of 2?

**Parameters**

| | | |
|---|---|---|
| in | _x | the number |

**Returns**

true if _x is a power of 2, false otherwise

#### 8.3.2.8 template<typename T > T gazebo::math::max ( const std::vector< T > & _values ) [inline]

get the maximum value of vector of values

**Parameters**

| | | |
|---|---|---|
| in | *_values* | the vector of values |

**Returns**

maximum

References gazebo::math::min().

Referenced by gazebo::math::clamp(), and gazebo::math::min().

#### 8.3.2.9 template<typename T > T gazebo::math::mean ( const std::vector< T > & *_values* ) [inline]

get mean of vector of values

**Parameters**

| | | |
|---|---|---|
| in | *_values* | the vector of values |

**Returns**

the mean

#### 8.3.2.10 template<typename T > T gazebo::math::min ( const std::vector< T > & *_values* ) [inline]

get the minimum value of vector of values

**Parameters**

| | | |
|---|---|---|
| in | *_values* | the vector of values |

**Returns**

minimum

References gazebo::math::max().

Referenced by gazebo::math::clamp(), and gazebo::math::max().

#### 8.3.2.11 double gazebo::math::parseFloat ( const std::string & *_input* ) [inline]

parse string into float

**Parameters**

| | |
|---|---|
| *_input* | the string |

**Returns**

a floating point number (can be NaN) or 0 with a message in the error stream

References gazebo::math::NAN_D.

#### 8.3.2.12 int gazebo::math::parseInt ( const std::string & *_input* ) [inline]

parse string into an integer

**Parameters**

| | | |
|---|---|---|
| in | *_input* | the string |

**Returns**

an integer, 0 or 0 and a message in the error stream

References gazebo::math::NAN_I.

#### 8.3.2.13 template<typename T > T gazebo::math::precision ( const T & *_a,* const unsigned int & *_precision* ) [inline]

get value at a specified precision

**Parameters**

| | | |
|---|---|---|
| in | *_a* | the number |
| in | *_precision* | the precision |

**Returns**

the value for the specified precision

#### 8.3.2.14 template<typename T > T gazebo::math::variance ( const std::vector< T > & *_values* ) [inline]

get variance of vector of values

**Parameters**

| | | |
|---|---|---|
| in | *_values* | the vector of values |

**Returns**

the squared deviation

### 8.3.3 Variable Documentation

#### 8.3.3.1 const double gazebo::math::NAN_D = std::numeric_limits<double>::quiet_NaN() [static]

Returns the representation of a quiet not a number (NAN)

Referenced by gazebo::math::parseFloat().

#### 8.3.3.2 const int gazebo::math::NAN_I = std::numeric_limits<int>::quiet_NaN() [static]

Returns the representation of a quiet not a number (NAN)

Referenced by gazebo::math::parseInt().

## 8.4 Messages

All messages and helper functions.

### Namespaces

- namespace **gazebo::msgs**

  *Messages namespace.*

### Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**

  *Google protobuf message generator for **gazebo::msgs*** (p. *108*).
- class **gazebo::msgs::MsgFactory**

  *A factory that generates protobuf message based on a string type.*

### Macros

- #define **GZ_REGISTER_STATIC_MSG**(_msgtype, _classname)

  *Static message registration macro.*

### Functions

- msgs::Vector3d **gazebo::msgs::Convert** (const math::Vector3 &_v)

  *Convert a **math::Vector3*** (p. *1091*) *to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion*** (p. *761*) *to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)

  *Convert a **math::Pose*** (p. *734*) *to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)

  *Convert a **common::Color*** (p. *233*) *to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

  *Convert a **common::Time*** (p. *1031*) *to a msgs::Time.*
- msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)

  *Convert a **math::Plane*** (p. *726*) *to a msgs::PlaneGeom.*
- math::Vector3 **gazebo::msgs::Convert** (const msgs::Vector3d &_v)

  *Convert a msgs::Vector3d to a math::Vector.*
- math::Quaternion **gazebo::msgs::Convert** (const msgs::Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion*** (p. *761*).
- math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose*** (p. *734*).
- common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color*** (p. *233*).
- common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time*** (p. *1031*).
- math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

*Convert a msgs::PlaneGeom to a common::Plane.*

- msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")

  *Create a request message.*
- msgs::Fog **gazebo::msgs::FogFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Fog from a fog SDF element.*
- msgs::Geometry **gazebo::msgs::GeometryFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Geometry from a geometry SDF element.*
- msgs::Header * **gazebo::msgs::GetHeader** (google::protobuf::Message &_message)

  *Get the header from a protobuf message.*
- msgs::GUI **gazebo::msgs::GUIFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::GUI from a GUI SDF element.*
- void **gazebo::msgs::Init** (google::protobuf::Message &_message, const std::string &_id="")

  *Initialize a message.*
- msgs::Light **gazebo::msgs::LightFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Light from a light SDF element.*
- msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::MeshGeom from a mesh SDF element.*
- msgs::Scene **gazebo::msgs::SceneFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Scene from a scene SDF element.*
- void **gazebo::msgs::Set** (common::Image &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 474).*
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 474).*
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 1091).*
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 1091).*
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 761).*
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 734).*
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 233).*
- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 1031).*
- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 726).*
- void **gazebo::msgs::Stamp** (msgs::Header *_header)

  *Time stamp a header.*
- void **gazebo::msgs::Stamp** (msgs::Time *_time)

  *Set the time in a time message.*
- msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::TrackVisual from a track visual SDF element.*
- msgs::Visual **gazebo::msgs::VisualFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Visual from a visual SDF element.*

### 8.4.1 Detailed Description

All messages and helper functions.

### 8.4.2 Macro Definition Documentation

#### 8.4.2.1 #define GZ_REGISTER_STATIC_MSG( _msgtype, _classname )

**Value:**

```
boost::shared_ptr<google::protobuf::Message> New##_classname() \
    { \
      return boost::shared_ptr<gazebo::msgs::_classname>(\
          new gazebo::msgs::_classname); \
    } \
    class Msg##_classname \
    { \
      public: Msg##_classname() \
      { \
        gazebo::msgs::MsgFactory::RegisterMsg(_msgtype, New##_classname);\
      } \
    }; \
    static Msg##_classname GzMsgInitializer;
```

Static message registration macro.

Use this macro to register messages.

**Parameters**

| | | |
|---|---|---|
| in | *_msgtype* | Message type name. |
| in | *_classname* | Class name for message. |

### 8.4.3 Function Documentation

#### 8.4.3.1 msgs::Vector3d gazebo::msgs::Convert ( const math::Vector3 & _v )

Convert a **math::Vector3** (p. 1091) to a msgs::Vector3d.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | The vector to convert |

**Returns**

A msgs::Vector3d object

#### 8.4.3.2 msgs::Quaternion gazebo::msgs::Convert ( const math::Quaternion & _q )

Convert a **math::Quaternion** (p. 761) to a msgs::Quaternion.

**Parameters**

| | | |
|---|---|---|
| in | *_q* | The quaternion to convert |

**Returns**

A msgs::Quaternion object

#### 8.4.3.3 msgs::Pose gazebo::msgs::Convert ( const math::Pose & _p )

Convert a **math::Pose** (p. 734) to a msgs::Pose.

**Parameters**

| | | |
|---|---|---|
| in | _p | The pose to convert |

**Returns**

A msgs::Pose object

#### 8.4.3.4 msgs::Color gazebo::msgs::Convert ( const common::Color & _c )

Convert a **common::Color** (p. 233) to a msgs::Color.

**Parameters**

| | | |
|---|---|---|
| in | _c | The color to convert |

**Returns**

A msgs::Color object

#### 8.4.3.5 msgs::Time gazebo::msgs::Convert ( const common::Time & _t )

Convert a **common::Time** (p. 1031) to a msgs::Time.

**Parameters**

| | | |
|---|---|---|
| in | _t | The time to convert |

**Returns**

A msgs::Time object

#### 8.4.3.6 msgs::PlaneGeom gazebo::msgs::Convert ( const math::Plane & _p )

Convert a **math::Plane** (p. 726) to a msgs::PlaneGeom.

**Parameters**

| | | |
|---|---|---|
| in | _p | The plane to convert |

**Returns**

A msgs::PlaneGeom object

#### 8.4.3.7 math::Vector3 gazebo::msgs::Convert ( const msgs::Vector3d & _v )

Convert a msgs::Vector3d to a math::Vector.

**Parameters**

| | | |
|---|---|---|
| in | _v | The plane to convert |

**Returns**

A **math::Vector3** (p. 1091) object

#### 8.4.3.8 math::Quaternion gazebo::msgs::Convert ( const msgs::Quaternion & _q )

Convert a msgs::Quaternion to a **math::Quaternion** (p. 761).

**Parameters**

| | | |
|---|---|---|
| in | _q | The quaternion to convert |

**Returns**

A **math::Quaternion** (p. 761) object

#### 8.4.3.9 math::Pose gazebo::msgs::Convert ( const msgs::Pose & _p )

Convert a msgs::Pose to a **math::Pose** (p. 734).

**Parameters**

| | | |
|---|---|---|
| in | _q | The pose to convert |

**Returns**

A **math::Pose** (p. 734) object

#### 8.4.3.10 common::Color gazebo::msgs::Convert ( const msgs::Color & _c )

Convert a msgs::Color to a **common::Color** (p. 233).

**Parameters**

| | | |
|---|---|---|
| in | _c | The color to convert |

**Returns**

A **common::Color** (p. 233) object

#### 8.4.3.11 common::Time gazebo::msgs::Convert ( const msgs::Time & _t )

Convert a msgs::Time to a **common::Time** (p. 1031).

**Parameters**

| | | |
|---|---|---|
| in | *_t* | The time to convert |

**Returns**

A **common::Time** (p. 1031) object

#### 8.4.3.12 math::Plane gazebo::msgs::Convert ( const msgs::PlaneGeom & _p )

Convert a msgs::PlaneGeom to a common::Plane.

**Parameters**

| | | |
|---|---|---|
| in | *_p* | The plane to convert |

**Returns**

A common::Plane object

#### 8.4.3.13 msgs::Request* gazebo::msgs::CreateRequest ( const std::string & _request, const std::string & _data = " " )

Create a request message.

**Parameters**

| | | |
|---|---|---|
| in | *_request* | Request string |
| in | *_data* | Optional data string |

**Returns**

A Request message

#### 8.4.3.14 msgs::Fog gazebo::msgs::FogFromSDF ( sdf::ElementPtr _sdf )

Create a msgs::Fog from a fog SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::Fog object

#### 8.4.3.15 msgs::Geometry gazebo::msgs::GeometryFromSDF ( sdf::ElementPtr _sdf )

Create a msgs::Geometry from a geometry SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::Geometry object

#### 8.4.3.16 msgs::Header* gazebo::msgs::GetHeader ( google::protobuf::Message & _message )

Get the header from a protobuf message.

**Parameters**

| | | |
|---|---|---|
| in | *_message* | A google protobuf message |

**Returns**

A pointer to the message's header

#### 8.4.3.17 msgs::GUI gazebo::msgs::GUIFromSDF ( sdf::ElementPtr _sdf )

Create a msgs::GUI from a GUI SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::GUI object

#### 8.4.3.18 void gazebo::msgs::Init ( google::protobuf::Message & _message, const std::string & _id = " " )

Initialize a message.

**Parameters**

| | | |
|---|---|---|
| in | *_message* | Message to initialize |
| in | *_id* | Optional string id |

Referenced by gazebo::physics::HingeJoint< SimbodyJoint >::Init().

#### 8.4.3.19 msgs::Light gazebo::msgs::LightFromSDF ( sdf::ElementPtr _sdf )

Create a msgs::Light from a light SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::Light object

#### 8.4.3.20 msgs::MeshGeom gazebo::msgs::MeshFromSDF ( sdf::ElementPtr _sdf )

Create a msgs::MeshGeom from a mesh SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::MeshGeom object

#### 8.4.3.21 msgs::Scene gazebo::msgs::SceneFromSDF ( sdf::ElementPtr _sdf )

Create a msgs::Scene from a scene SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::Scene object

#### 8.4.3.22 void gazebo::msgs::Set ( common::Image & _img, const msgs::Image & _msg )

Convert a msgs::Image to a **common::Image** (p. 474).

**Parameters**

| | | |
|---|---|---|
| out | *_img* | The **common::Image** (p. 474) container |
| in | *_msg* | The Image message to convert |

#### 8.4.3.23 void gazebo::msgs::Set ( msgs::Image * _msg, const common::Image & _i )

Set a msgs::Image from a **common::Image** (p. 474).

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | A msgs::Image pointer |
| in | *_i* | A **common::Image** (p. 474) reference |

#### 8.4.3.24 void gazebo::msgs::Set ( msgs::Vector3d * _pt, const math::Vector3 & _v )

Set a msgs::Vector3d from a **math::Vector3** (p. 1091).

**Parameters**

| | | |
|---|---|---|
| out | *_pt* | A msgs::Vector3d pointer |
| in | *_v* | A **math::Vector3** (p. 1091) reference |

#### 8.4.3.25 void gazebo::msgs::Set ( msgs::Vector2d * _pt, const math::Vector2d & _v )

Set a msgs::Vector2d from a **math::Vector3** (p. 1091).

**Parameters**

| | | |
|---|---|---|
| out | *_pt* | A msgs::Vector2d pointer |
| in | *_v* | A **math::Vector2d** (p. 1074) reference |

#### 8.4.3.26 void gazebo::msgs::Set ( msgs::Quaternion * _q, const math::Quaternion & _v )

Set a msgs::Quaternion from a **math::Quaternion** (p. 761).

**Parameters**

| | | |
|---|---|---|
| out | *_q* | A msgs::Quaternion pointer |
| in | *_v* | A **math::Quaternion** (p. 761) reference |

#### 8.4.3.27 void gazebo::msgs::Set ( msgs::Pose * _p, const math::Pose & _v )

Set a msgs::Pose from a **math::Pose** (p. 734).

**Parameters**

| | | |
|---|---|---|
| out | *_p* | A msgs::Pose pointer |
| in | *_v* | A **math::Pose** (p. 734) reference |

#### 8.4.3.28 void gazebo::msgs::Set ( msgs::Color * _c, const common::Color & _v )

Set a msgs::Color from a **common::Color** (p. 233).

**Parameters**

| | | |
|---|---|---|
| out | *_p* | A msgs::Color pointer |
| in | *_v* | A **common::Color** (p. 233) reference |

#### 8.4.3.29 void gazebo::msgs::Set ( msgs::Time * _t, const common::Time & _v )

Set a msgs::Time from a **common::Time** (p. 1031).

**Parameters**

| | | |
|---|---|---|
| out | *_p* | A msgs::Time pointer |
| in | *_v* | A **common::Time** (p. 1031) reference |

#### 8.4.3.30 void gazebo::msgs::Set ( msgs::PlaneGeom * _p, const math::Plane & _v )

Set a msgs::Plane from a **math::Plane** (p. 726).

**Parameters**

| | | |
|---|---|---|
| out | *_p* | A msgs::Plane pointer |
| in | *_v* | A **math::Plane** (p. 726) reference |

#### 8.4.3.31 void gazebo::msgs::Stamp ( msgs::Header * _header )

Time stamp a header.

**Parameters**

| | | |
|---|---|---|
| in | *_header* | Header to stamp |

#### 8.4.3.32 void gazebo::msgs::Stamp ( msgs::Time * _time )

Set the time in a time message.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | A Time message |

#### 8.4.3.33 msgs::TrackVisual gazebo::msgs::TrackVisualFromSDF ( sdf::ElementPtr _sdf )

Create a msgs::TrackVisual from a track visual SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::TrackVisual object

#### 8.4.3.34 msgs::Visual gazebo::msgs::VisualFromSDF ( sdf::ElementPtr *_sdf* )

Create a msgs::Visual from a visual SDF element.

**Parameters**

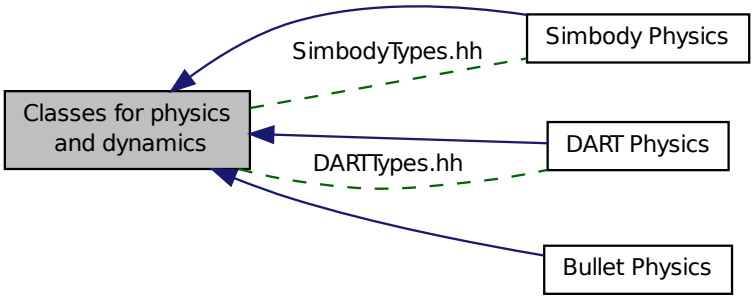
| | | |
|---|---|---|
| in | *_sdf* | The sdf element |

**Returns**

The new msgs::Visual object

## 8.5 Classes for physics and dynamics

Collaboration diagram for Classes for physics and dynamics:




### Modules

- **DART Physics**

  *dart physics engine wrapper*
- **Bullet Physics**
- **Simbody Physics**

  *simbody physics engine wrapper*

### Files

- file **DARTTypes.hh**

  *DART wrapper forward declarations and typedefs.*
- file **PhysicsTypes.hh**

  *default namespace for gazebo*
- file **SimbodyTypes.hh**

  *Simbody wrapper forward declarations and typedefs.*

### Namespaces

- namespace **gazebo::physics**

  *namespace for physics*

### Classes

- class **gazebo::physics::Actor**

  ***Actor*** (p. *131*) *class enables GPU based mesh model / skeleton scriptable animation.*
- class **gazebo::physics::BallJoint**< **T** >

***Base*** *(p. 159) class for a ball joint.*

- class **gazebo::physics::Base**

  ***Base*** *(p. 159) class for most physics classes.*

- class **gazebo::physics::BoxShape**

  *Box geometry primitive.*

- class **gazebo::physics::Collision**

  ***Base*** *(p. 159) class for all collision entities.*

- class **gazebo::physics::CollisionState**

  *Store state information of a* ***physics::Collision*** *(p. 220) object.*

- class **gazebo::physics::Contact**

  *A contact between two collisions.*

- class **gazebo::physics::ContactManager**

  *Aggregates all the contact information generated by the collision detection engine.*

- class **gazebo::physics::CylinderShape**

  *Cylinder collision.*

- class **gazebo::physics::Entity**

  ***Base*** *(p. 159) class for all physics objects in Gazebo.*

- class **gazebo::physics::Gripper**

  *A gripper abstraction.*

- class **gazebo::physics::HeightmapShape**

  ***HeightmapShape*** *(p. 465) collision shape builds a heightmap from an image.*

- class **gazebo::physics::Hinge2Joint**< **T** >

  *A two axis hinge joint.*

- class **gazebo::physics::HingeJoint**< **T** >

  *A single axis hinge joint.*

- class **gazebo::physics::Inertial**

  *A class for inertial information about a link.*

- class **gazebo::physics::Joint**

  ***Base*** *(p. 159) class for all joints.*

- class **gazebo::physics::JointController**

  *A class for manipulating* ***physics::Joint*** *(p. 496).*

- class **gazebo::physics::JointState**

  *keeps track of state of a* ***physics::Joint*** *(p. 496)*

- class **gazebo::physics::JointWrench**

  *Wrench information from a joint.*

- class **gazebo::physics::Link**

  ***Link*** *(p. 542) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*

- class **gazebo::physics::LinkState**

  *Store state information of a* ***physics::Link*** *(p. 542) object.*

- class **Logplay**

  *Open and playback log files that were recorded using LogRecord.*

- class **gazebo::util::LogPlay**

- class **gazebo::physics::MapShape**

  *Creates box extrusions based on an image.*

- class **gazebo::physics::MeshShape**

  *Triangle mesh collision shape.*

- class **gazebo::physics::Model**

  *A model is a collection of links, joints, and plugins.*
- class **gazebo::physics::ModelState**

  *Store state information of a* ***physics::Model*** *(*p. *624) object.*
- class **gazebo::physics::MultiRayShape**

  *Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.*
- class **gazebo::physics::PhysicsEngine**

  ***Base*** *(*p. *159) class for a physics engine.*
- class **gazebo::physics::PhysicsFactory**

  *The physics factory instantiates different physics engines.*
- class **gazebo::physics::PlaneShape**

  ***Collision*** *(*p. *220) for an infinite plane.*
- class **QuadNode**
- class **gazebo::physics::RayShape**

  ***Base*** *(*p. *159) class for Ray collision geometry.*
- class **gazebo::physics::Road**

  *for building a* ***Road*** *(*p. *804) from SDF*
- class **gazebo::physics::ScrewJoint**< **T** >

  *A screw joint, which has both prismatic and rotational DOFs.*
- class **gazebo::physics::Shape**

  ***Base*** *(*p. *159) class for all shapes.*
- class **gazebo::physics::SimbodyModel**

  *A model is a collection of links, joints, and plugins.*
- class **gazebo::physics::SliderJoint**< **T** >

  *A slider joint.*
- class **gazebo::physics::SphereShape**

  *Sphere collision shape.*
- class **gazebo::physics::State**

  ***State*** *(*p. *998) of an entity.*
- class **gazebo::physics::SurfaceParams**

  ***SurfaceParams*** *(*p. *1020) defines various Surface contact parameters.*
- class **gazebo::physics::UniversalJoint**< **T** >

  *A universal joint.*
- class **gazebo::physics::World**

  *The world provides access to all other object within a simulated environment.*
- class **gazebo::physics::WorldState**

  *Store state information of a* ***physics::World*** *(*p. *1157) object.*

## Macros

- #define **GZ_REGISTER_PHYSICS_ENGINE**(name, classname)

  *Static physics registration macro.*

## Typedefs

- typedef PhysicsEnginePtr(∗ **gazebo::physics::PhysicsFactoryFn** )(WorldPtr world)

## Functions

- WorldPtr **gazebo::physics::create_world** (const std::string &_name="")

  *Create a world given a name.*
- bool **gazebo::physics::fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 94).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")

  *Returns a pointer to a world by name.*
- uint32_t **gazebo::physics::getUniqueId** ()

  *Get a unique ID.*
- void **gazebo::physics::init_world** (WorldPtr _world)

  *Init world given a pointer to it.*
- void **gazebo::physics::init_worlds** ()

  *initialize multiple worlds stored in static variable gazebo::g_worlds*
- bool **gazebo::physics::load** ()

  *Setup **gazebo::SystemPlugin** (p. 1030)'s and call **gazebo::transport::init** (p. 95).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)

  *Load world from sdf::Element pointer.*
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)

  *load multiple worlds from single sdf::Element pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 1167).*
- void **gazebo::physics::pause_worlds** (bool pause)

  *pause multiple worlds stored in static variable gazebo::g_worlds*
- void **gazebo::physics::remove_worlds** ()

  *remove multiple worlds stored in static variable gazebo::g_worlds*
- void **gazebo::physics::run_world** (WorldPtr _world, unsigned int _iterations=0)

  *Run world by calling **World::Run()** (p. 1167) given a pointer to it.*
- void **gazebo::physics::run_worlds** (unsigned int _iterations=0)

  *Run multiple worlds stored in static variable gazebo::g_worlds.*
- void **gazebo::physics::stop_world** (WorldPtr _world)

  *Stop world by calling **World::Stop()** (p. 1168) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()

  *stop multiple worlds stored in static variable gazebo::g_worlds*
- bool **gazebo::physics::worlds_running** ()

  *Return true if any world is running.*

## Variables

- static std::string **gazebo::physics::EntityTypename** [ ]

  *String names for the different entity types.*

### 8.5.1 Detailed Description

### 8.5.2 Macro Definition Documentation

#### 8.5.2.1 #define GZ_REGISTER_PHYSICS_ENGINE( *name, classname* )

**Value:**

```
PhysicsEnginePtr New##classname(WorldPtr _world) \
    { \
      return PhysicsEnginePtr(new gazebo::physics::classname(_world)); \
    } \
    void Register##classname() \
    {\
      PhysicsFactory::RegisterPhysicsEngine(name, New##classname);\
    }
```

Static physics registration macro.

Use this macro to register physics engine with the server.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Physics type name, as it appears in the world file. |
| in | *classname* | C++ class name for the physics engine. |

### 8.5.3 Typedef Documentation

#### 8.5.3.1 typedef PhysicsEnginePtr(∗ gazebo::physics::PhysicsFactoryFn)(WorldPtr world)

### 8.5.4 Function Documentation

#### 8.5.4.1 WorldPtr gazebo::physics::create_world ( const std::string & *_name* = " " )

Create a world given a name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the world to create. |

**Returns**

Pointer to the new world.

#### 8.5.4.2 bool gazebo::physics::fini ( )

Finalize transport by calling **gazebo::transport::fini** (p. 94).

#### 8.5.4.3 WorldPtr gazebo::physics::get_world ( const std::string & *_name* = " " )

Returns a pointer to a world by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the world to get. |

**Returns**

Pointer to the world.

Referenced by Joint_TEST::SpawnJoint().

#### 8.5.4.4 uint32_t gazebo::physics::getUniqueId ( )

Get a unique ID.

**Returns**

A unique integer

#### 8.5.4.5 void gazebo::physics::init_world ( WorldPtr *_world* )

Init world given a pointer to it.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | **World** (p. 1157) to initialize. |

#### 8.5.4.6 void gazebo::physics::init_worlds ( )

initialize multiple worlds stored in static variable gazebo::g_worlds

#### 8.5.4.7 bool gazebo::physics::load ( )

Setup **gazebo::SystemPlugin** (p. 1030)'s and call **gazebo::transport::init** (p. 95).

#### 8.5.4.8 void gazebo::physics::load_world ( WorldPtr *_world,* sdf::ElementPtr *_sdf* )

Load world from sdf::Element pointer.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to a world. |
| in | *_sdf* | SDF values to load from. |

#### 8.5.4.9 void gazebo::physics::load_worlds ( sdf::ElementPtr *_sdf* )

load multiple worlds from single sdf::Element pointer

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values used to create worlds. |

#### 8.5.4.10 void gazebo::physics::pause_world ( WorldPtr *_world,* bool *_pause* )

Pause world by calling **World::SetPaused** (p. 1167).

**Parameters**

| | | |
|---|---|---|
| in | *_world* | **World** (p. 1157) to pause or unpause. |
| in | *_pause* | True to pause, False to unpause. |

#### 8.5.4.11 void gazebo::physics::pause_worlds ( bool *pause* )

pause multiple worlds stored in static variable gazebo::g_worlds

**Parameters**

| | | |
|---|---|---|
| in | *_pause* | True to pause, False to unpause. |

#### 8.5.4.12 void gazebo::physics::remove_worlds ( )

remove multiple worlds stored in static variable gazebo::g_worlds

#### 8.5.4.13 void gazebo::physics::run_world ( WorldPtr *_world,* unsigned int *_iterations* = 0 )

Run world by calling **World::Run()** (p. 1167) given a pointer to it.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | **World** (p. 1157) to run. |
| in | *_iterations* | Number of iterations for each world to take. Zero indicates that each world should continue forever. |

#### 8.5.4.14 void gazebo::physics::run_worlds ( unsigned int *_iterations* = 0 )

Run multiple worlds stored in static variable gazebo::g_worlds.

**Parameters**

| | | |
|---|---|---|
| in | *_iterations* | Number of iterations for each world to take. Zero indicates that each world should continue forever. |

#### 8.5.4.15 void gazebo::physics::stop_world ( WorldPtr *_world* )

Stop world by calling **World::Stop()** (p. 1168) given a pointer to it.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | **World** (p. 1157) to stop. |

#### 8.5.4.16 void gazebo::physics::stop_worlds ( )

stop multiple worlds stored in static variable gazebo::g_worlds

#### 8.5.4.17 bool gazebo::physics::worlds_running ( )

Return true if any world is running.

**Returns**

True if any world is running.

### 8.5.5 Variable Documentation

#### 8.5.5.1 std::string gazebo::physics::EntityTypename[] [static]

**Initial value:**

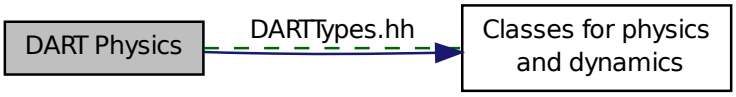
```
= {
      "common",
      "entity",
      "model",
      "actor",
      "link",
      "collision",
      "light",
      "visual",
      "joint",
      "ball",
      "hinge2",
      "hinge",
      "slider",
      "universal",
      "shape",
      "box",
      "cylinder",
      "heightmap",
      "map",
      "multiray",
      "ray",
      "plane",
      "sphere",
      "trimesh"
    }
```

String names for the different entity types.

## 8.6 DART Physics

dart physics engine wrapper

Collaboration diagram for DART Physics:




### Files

- file **DARTTypes.hh**

  *DART wrapper forward declarations and typedefs.*

### Classes

- class **gazebo::physics::DARTLink**

  *DART **Link** (p. 542) class.*
- class **gazebo::physics::DARTModel**

  *DART model class.*
- class **gazebo::physics::DARTPhysics**

  *DART physics engine.*
- class **gazebo::physics::DARTRayShape**

  *Ray collision.*
- class **gazebo::physics::DARTTypes**

  *A set of functions for converting between the math types used by gazebo and dart.*

### Functions

- **gazebo::physics::DARTRayShape::DARTRayShape** (PhysicsEnginePtr _physicsEngine)

  *Constructor for a global ray.*
- **gazebo::physics::DARTRayShape::DARTRayShape** (CollisionPtr _collision)

  *Constructor.*
- virtual **gazebo::physics::DARTRayShape::~DARTRayShape** ()

  *Destructor.*
- static Eigen::Isometry3d **gazebo::physics::DARTTypes::ConvPose** (const math::Pose &_pose)
- static math::Pose **gazebo::physics::DARTTypes::ConvPose** (const Eigen::Isometry3d &_T)
- static Eigen::Quaterniond **gazebo::physics::DARTTypes::ConvQuat** (const math::Quaternion &_quat)
- static math::Quaternion **gazebo::physics::DARTTypes::ConvQuat** (const Eigen::Quaterniond &_quat)
- static Eigen::Vector3d **gazebo::physics::DARTTypes::ConvVec3** (const math::Vector3 &_vec3)
- static math::Vector3 **gazebo::physics::DARTTypes::ConvVec3** (const Eigen::Vector3d &_vec3)

- virtual void **gazebo::physics::DARTRayShape::GetIntersection** (double &_dist, std::string &_entity)

  *Get the nearest intersection.*
- virtual void **gazebo::physics::DARTRayShape::SetPoints** (const math::Vector3 &_posStart, const math::Vector3 &_posEnd)

  *Set the ray based on starting and ending points relative to the body.*
- virtual void **gazebo::physics::DARTRayShape::Update** ()

  *Update the ray collision.*

### 8.6.1 Detailed Description

dart physics engine wrapper

### 8.6.2 Function Documentation

#### 8.6.2.1 gazebo::physics::DARTRayShape::DARTRayShape ( PhysicsEnginePtr *_physicsEngine* ) [explicit]

Constructor for a global ray.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Pointer to the physics engine. |

#### 8.6.2.2 gazebo::physics::DARTRayShape::DARTRayShape ( CollisionPtr *_collision* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_collision* | **Collision** (p. 220) object this ray is attached to. |

#### 8.6.2.3 virtual gazebo::physics::DARTRayShape::~DARTRayShape ( ) [virtual]

Destructor.

#### 8.6.2.4 static Eigen::Isometry3d gazebo::physics::DARTTypes::ConvPose ( const math::Pose & *_pose* ) [inline], [static]

References gazebo::physics::DARTTypes::ConvQuat(), gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::pos, and gazebo::math::Pose::rot.

#### 8.6.2.5 static math::Pose gazebo::physics::DARTTypes::ConvPose ( const Eigen::Isometry3d & *_T* ) [inline], [static]

References gazebo::physics::DARTTypes::ConvQuat(), gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::pos, and gazebo::math::Pose::rot.

#### 8.6.2.6 static Eigen::Quaterniond gazebo::physics::DARTTypes::ConvQuat ( const math::Quaternion & *_quat* ) [inline], [static]

References gazebo::math::Quaternion::w, gazebo::math::Quaternion::x, gazebo::math::Quaternion::y, and gazebo::math::Quaternion::z.

Referenced by gazebo::physics::DARTTypes::ConvPose().

#### 8.6.2.7 static math::Quaternion gazebo::physics::DARTTypes::ConvQuat ( const Eigen::Quaterniond & *_quat* ) [inline], [static]

#### 8.6.2.8 static Eigen::Vector3d gazebo::physics::DARTTypes::ConvVec3 ( const math::Vector3 & *_vec3* ) [inline], [static]

References gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

Referenced by gazebo::physics::DARTTypes::ConvPose(), and gazebo::physics::DARTBoxShape::SetSize().

#### 8.6.2.9 static math::Vector3 gazebo::physics::DARTTypes::ConvVec3 ( const Eigen::Vector3d & *_vec3* ) [inline], [static]

#### 8.6.2.10 virtual void gazebo::physics::DARTRayShape::GetIntersection ( double & *_dist,* std::string & *_entity* ) [virtual]

Get the nearest intersection.

**Parameters**

| | | |
|---|---|---|
| out | *_dist* | Distance to the intersection. |
| out | *_entity* | Name of the entity that was hit. |

Implements **gazebo::physics::RayShape** (p. 789).

#### 8.6.2.11 virtual void gazebo::physics::DARTRayShape::SetPoints ( const math::Vector3 & *_posStart,* const math::Vector3 & *_posEnd* ) [virtual]

Set the ray based on starting and ending points relative to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_posStart* | Start position, relative the body |
| in | *_posEnd* | End position, relative to the body |

Reimplemented from **gazebo::physics::RayShape** (p. 790).

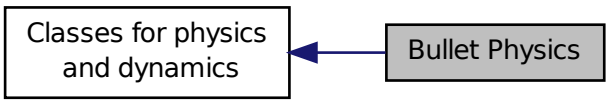
#### 8.6.2.12 virtual void gazebo::physics::DARTRayShape::Update ( ) [virtual]

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 791).

## 8.7 Bullet Physics

Collaboration diagram for Bullet Physics:




### Classes

- class **gazebo::physics::DARTMultiRayShape**

  *DART specific version of **MultiRayShape** (p. 664).*

### Functions

- **gazebo::physics::DARTMultiRayShape::DARTMultiRayShape** (CollisionPtr _parent)

  *Constructor.*
- virtual **gazebo::physics::DARTMultiRayShape::∼DARTMultiRayShape** ()

  *Destructor.*
- void **gazebo::physics::DARTMultiRayShape::AddRay** (const math::Vector3 &_start, const math::Vector3 &_end)

  *Add a ray to the collision.*
- virtual void **gazebo::physics::DARTMultiRayShape::UpdateRays** ()

  *Physics engine specific method for updating the rays.*

### 8.7.1 Detailed Description

### 8.7.2 Function Documentation

#### 8.7.2.1 gazebo::physics::DARTMultiRayShape::DARTMultiRayShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent **Collision** (p. 220). |

#### 8.7.2.2 virtual gazebo::physics::DARTMultiRayShape::∼DARTMultiRayShape ( ) [virtual]

Destructor.

#### 8.7.2.3 void gazebo::physics::DARTMultiRayShape::AddRay ( const math::Vector3 & _start, const math::Vector3 & _end ) [protected], [virtual]

Add a ray to the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_start* | Start location of the ray. |
| in | *_end* | End location of the ray. |

Reimplemented from **gazebo::physics::MultiRayShape** (p. 667).

#### 8.7.2.4 virtual void gazebo::physics::DARTMultiRayShape::UpdateRays ( ) [virtual]

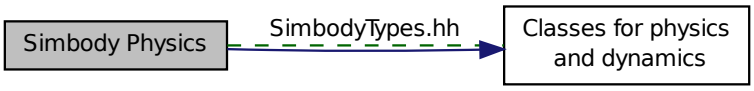
Physics engine specific method for updating the rays.

Implements **gazebo::physics::MultiRayShape** (p. 671).

## 8.8 Simbody Physics

simbody physics engine wrapper

Collaboration diagram for Simbody Physics:




### Files

- file **SimbodyTypes.hh**

  *Simbody wrapper forward declarations and typedefs.*

### Classes

- class **gazebo::physics::SimbodyBallJoint**

  ***SimbodyBallJoint*** *(*p. *865) class models a ball joint in Simbody.*
- class **gazebo::physics::SimbodyBoxShape**

  *Simbody box collision.*
- class **gazebo::physics::SimbodyCollision**

  *Simbody collisions.*
- class **gazebo::physics::SimbodyCylinderShape**

  *Cylinder collision.*
- class **gazebo::physics::SimbodyHeightmapShape**

  *Height map collision.*
- class **gazebo::physics::SimbodyHinge2Joint**

  *A two axis hinge joint.*
- class **gazebo::physics::SimbodyHingeJoint**

  *A single axis hinge joint.*
- class **gazebo::physics::SimbodyJoint**

  ***Base*** *(*p. *159) class for all joints.*
- class **gazebo::physics::SimbodyLink**

  *Simbody* ***Link*** *(*p. *542) class.*
- class **gazebo::physics::SimbodyMeshShape**

  *Triangle mesh collision.*
- class **gazebo::physics::SimbodyMultiRayShape**

  *Simbody specific version of* ***MultiRayShape*** *(*p. *664).*
- class **gazebo::physics::SimbodyPhysics**

  *Simbody physics engine.*
- class **gazebo::physics::SimbodyPlaneShape**

*Simbody collision for an infinite plane.*

- class **gazebo::physics::SimbodyRayShape**

  *Ray shape for simbody.*
- class **gazebo::physics::SimbodyScrewJoint**

  *A screw joint.*
- class **gazebo::physics::SimbodySliderJoint**

  *A slider joint.*
- class **gazebo::physics::SimbodySphereShape**

  *Simbody sphere collision.*
- class **gazebo::physics::SimbodyUniversalJoint**

  *A simbody universal joint class.*

### 8.8.1 Detailed Description

simbody physics engine wrapper

## 8.9 Rendering

A set of rendering related class, functions, and definitions.

### Namespaces

- namespace **gazebo::rendering**

  *Rendering namespace.*

### Classes

- class **gazebo::rendering::ArrowVisual**

  *Basic arrow visualization.*
- class **gazebo::rendering::AxisVisual**

  *Basic axis visualization.*
- class **gazebo::rendering::Camera**

  *Basic camera sensor.*
- class **gazebo::rendering::CameraVisual**

  *Basic camera visualization.*
- class **gazebo::rendering::COMVisual**

  *Basic Center of Mass visualization.*
- class **gazebo::rendering::ContactVisual**

  *Contact visualization.*
- class **gazebo::rendering::Conversions**

  ***Conversions*** *(*p. *273)* ***Conversions.hh*** *(*p. *1218)* ***rendering/Conversions.hh*** *(*p. *1218).*
- class **gazebo::rendering::DepthCamera**

  *Depth camera used to render depth data into an image buffer.*
- class **gazebo::rendering::DynamicLines**

  *Class for drawing lines that can change.*
- class **gazebo::rendering::DynamicRenderable**

  *Abstract base class providing mechanisms for dynamically growing hardware buffers.*
- class **gazebo::rendering::Events**

  *Base class for rendering events.*
- class **gazebo::rendering::FPSViewController**

  *First Person Shooter style view controller.*
- class **gazebo::rendering::GpuLaser**

  *GPU based laser distance sensor.*
- class **gazebo::rendering::Grid**

  *Displays a grid of cells, drawn with lines.*
- class **gazebo::rendering::GUIOverlay**

  *A class that creates a CEGUI overlay on a render window.*
- class **gazebo::rendering::Heightmap**

  *Rendering a terrain using heightmap information.*
- class **gazebo::rendering::JointVisual**

  *Visualization for joints.*
- class **gazebo::rendering::LaserVisual**

*Visualization for laser data.*

- class **gazebo::rendering::Light**

  *A light source.*

- class **gazebo::rendering::MovableText**

  *Movable text.*

- class **gazebo::rendering::OrbitViewController**

  *Orbit view controller.*

- class **gazebo::rendering::Projector**

  *Projects a material onto surface, light a light projector.*

- class **gazebo::rendering::RenderEngine**

  *Adaptor to Ogre3d.*

- class **gazebo::rendering::RFIDTagVisual**

  *Visualization for RFID tags sensor.*

- class **gazebo::rendering::RFIDVisual**

  *Visualization for RFID sensor.*

- class **Road**

  *Used to render a strip of road.*

- class **gazebo::rendering::Road2d**
- class **gazebo::rendering::RTShaderSystem**

  *Implements **Ogre** (p. 130)'s Run-Time Shader system.*

- class **gazebo::rendering::Scene**

  *Representation of an entire scene graph.*

- class **gazebo::rendering::SelectionObj**

  *Interactive selection object for models and links.*

- class **gazebo::rendering::SonarVisual**

  *Visualization for sonar data.*

- class **gazebo::rendering::TransmitterVisual**

  *Visualization for the wireless propagation data.*

- class **gazebo::rendering::UserCamera**

  *A camera used for user visualization of a scene.*

- class **gazebo::rendering::VideoVisual**

  *A visual element that displays a video as a texture.*

- class **gazebo::rendering::ViewController**

  *Base class for view controllers.*

- class **gazebo::rendering::Visual**

  *A renderable object.*

- class **gazebo::rendering::WindowManager**

  *Class to mangage render windows.*

- class **gazebo::rendering::WireBox**

  *Draws a wireframe box.*

- class **gazebo::rendering::WrenchVisual**

  *Visualization for sonar data.*

## Enumerations

- enum **gazebo::rendering::SelectionObj::SelectionMode** {
  **gazebo::rendering::SelectionObj::SELECTION_NONE** = 0, **gazebo::rendering::SelectionObj::TRANS**, **gazebo::rendering::SelectionObj::ROT**, **gazebo::rendering::SelectionObj::SCALE**, **gazebo::rendering::SelectionObj::TRANS_X**, **gazebo::rendering::SelectionObj::TRANS_Y**, **gazebo::rendering::SelectionObj::TRANS_Z**, **gazebo::rendering::SelectionObj::ROT_X**, **gazebo::rendering::SelectionObj::ROT_Y**, **gazebo::rendering::SelectionObj::ROT_Z**, **gazebo::rendering::SelectionObj::SCALE_X**, **gazebo::rendering::SelectionObj::SCALE_Y**, **gazebo::rendering::SelectionObj::SCALE_Z** }

## Functions

- **gazebo::rendering::SelectionObj::SelectionObj** (const std::string &_name, VisualPtr _vis)

  *Constructor.*
- virtual **gazebo::rendering::SelectionObj::**∼**SelectionObj** ()

  *Deconstructor.*
- void **gazebo::rendering::SelectionObj::Attach** (rendering::VisualPtr _vis)

  *Attach the selection object to the given visual.*
- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)

  *create **rendering::Scene** (p. 814) by name.*
- void **gazebo::rendering::SelectionObj::Detach** ()

  *Detach the selection object from the current visual.*
- bool **gazebo::rendering::fini** ()

  *teardown rendering engine.*
- rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")

  *get pointer to **rendering::Scene** (p. 814) by name.*
- SelectionMode **gazebo::rendering::SelectionObj::GetMode** ()

  *Get the current selection mode.*
- SelectionMode **gazebo::rendering::SelectionObj::GetState** ()

  *Get the current selection state.*
- bool **gazebo::rendering::init** ()

  *init rendering engine.*
- bool **gazebo::rendering::load** ()

  *load rendering engine.*
- void **gazebo::rendering::SelectionObj::Load** ()

  *Load.*
- void **gazebo::rendering::remove_scene** (const std::string &_name)

  *remove a **rendering::Scene** (p. 814) by name*
- void **gazebo::rendering::SelectionObj::SetGlobal** (bool _global)

  *Set selection object to ignore local transforms.*
- void **gazebo::rendering::SelectionObj::SetMode** (const std::string &_mode)

  *Set the manipulation mode.*
- void **gazebo::rendering::SelectionObj::SetMode** (SelectionMode _mode)

  *Set the selection mode.*
- void **gazebo::rendering::SelectionObj::SetState** (const std::string &_state)

  *Set state by highlighting the corresponding selection object visual.*

- void **gazebo::rendering::SelectionObj::SetState** (SelectionMode _state)

  *Set state by highlighting the corresponding selection object visual.*
- void **gazebo::rendering::SelectionObj::UpdateSize** ()

  *Update selection object size to match the parent visual.*

### 8.9.1 Detailed Description

A set of rendering related class, functions, and definitions.

### 8.9.2 Enumeration Type Documentation

#### 8.9.2.1 enum gazebo::rendering::SelectionObj::SelectionMode

**Enumerator**

***SELECTION_NONE*** Translation in x.

***TRANS*** Translation mode.

***ROT*** Rotation mode.

***SCALE*** Scale mode.

***TRANS_X*** Translation in x.

***TRANS_Y*** Translation in y.

***TRANS_Z*** Translation in z.

***ROT_X*** Rotation in x.

***ROT_Y*** Rotation in y.

***ROT_Z*** Rotation in z.

***SCALE_X*** Scale in x.

***SCALE_Y*** Scale in y.

***SCALE_Z*** Scale in z.

### 8.9.3 Function Documentation

#### 8.9.3.1 gazebo::rendering::SelectionObj::SelectionObj ( const std::string & *_name,* VisualPtr *_vis* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of selection object. |
| in | *_vis* | Parent visual that the selection object is attached to. |

#### 8.9.3.2 virtual gazebo::rendering::SelectionObj::~SelectionObj ( ) [virtual]

Deconstructor.

#### 8.9.3.3 void gazebo::rendering::SelectionObj::Attach ( rendering::VisualPtr _vis )

Attach the selection object to the given visual.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | Pointer to visual to which the selection object will be attached. |

#### 8.9.3.4 rendering::ScenePtr gazebo::rendering::create_scene ( const std::string & _name, bool _enableVisualizations, bool _isServer = false )

create **rendering::Scene** (p. 814) by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the scene to create. |
| in | *_enable-Visualizations* | True enables visualization elements such as laser lines. |

#### 8.9.3.5 void gazebo::rendering::SelectionObj::Detach ( )

Detach the selection object from the current visual.

#### 8.9.3.6 bool gazebo::rendering::fini ( )

teardown rendering engine.

#### 8.9.3.7 rendering::ScenePtr gazebo::rendering::get_scene ( const std::string & _name = " " )

get pointer to **rendering::Scene** (p. 814) by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the scene to retreive. |

#### 8.9.3.8 SelectionMode gazebo::rendering::SelectionObj::GetMode ( )

Get the current selection mode.

#### 8.9.3.9 SelectionMode gazebo::rendering::SelectionObj::GetState ( )

Get the current selection state.

#### 8.9.3.10 bool gazebo::rendering::init ( )

init rendering engine.

#### 8.9.3.11 bool gazebo::rendering::load ( )

load rendering engine.

#### 8.9.3.12 void gazebo::rendering::SelectionObj::Load ( ) [virtual]

Load.

Reimplemented from **gazebo::rendering::Visual** (p. 1135).

#### 8.9.3.13 void gazebo::rendering::remove_scene ( const std::string & *_name* )

remove a **rendering::Scene** (p. 814) by name

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the scene to remove. |

#### 8.9.3.14 void gazebo::rendering::SelectionObj::SetGlobal ( bool *_global* )

Set selection object to ignore local transforms.

**Parameters**

| | | |
|---|---|---|
| in | *_global* | True to set the visuals to be in global frame. |

#### 8.9.3.15 void gazebo::rendering::SelectionObj::SetMode ( const std::string & *_mode* )

Set the manipulation mode.

**Parameters**

| | | |
|---|---|---|
| in | *_mode* | Manipulation mode in string: translate rotate, scale. |

#### 8.9.3.16 void gazebo::rendering::SelectionObj::SetMode ( SelectionMode *_mode* )

Set the selection mode.

_name Selection mode: TRANS, ROT, SCALE.

#### 8.9.3.17 void gazebo::rendering::SelectionObj::SetState ( const std::string & *_state* )

Set state by highlighting the corresponding selection object visual.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | Selection state in string format. |

#### 8.9.3.18 void gazebo::rendering::SelectionObj::SetState ( SelectionMode _state )

Set state by highlighting the corresponding selection object visual.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | Selection state. |

**See Also**

**SelectionMode** (p. 83)

#### 8.9.3.19 void gazebo::rendering::SelectionObj::UpdateSize ( )

Update selection object size to match the parent visual.

## 8.10 Sensors

A set of sensor classes, functions, and definitions.

### Files

- file **SensorTypes.hh**

  *Forward declarations and typedefs for sensors.*

### Namespaces

- namespace **gazebo::sensors**

  *Sensors namespace.*

### Classes

- class **gazebo::sensors::CameraSensor**

  *Basic camera sensor.*
- class **gazebo::sensors::ContactSensor**

  *Contact sensor.*
- class **gazebo::sensors::DepthCameraSensor**
- class **gazebo::sensors::ForceTorqueSensor**

  ***Sensor*** *(*p. *837) for measure force and torque on a joint.*
- class **gazebo::sensors::GpsSensor**

  ***GpsSensor*** *(*p. *426) to provide position measurement.*
- class **gazebo::sensors::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**

  *An IMU sensor.*
- class **gazebo::sensors::MultiCameraSensor**

  *Multiple camera sensor.*
- class **gazebo::sensors::Noise**

  ***Noise*** *(*p. *689) models for sensor output signals.*
- class **gazebo::sensors::RaySensor**

  ***Sensor*** *(*p. *837) with one or more rays.*
- class **gazebo::sensors::RFIDSensor**

  ***Sensor*** *(*p. *837) class for RFID type of sensor.*
- class **gazebo::sensors::RFIDTag**

  ***RFIDTag*** *(*p. *798) to interact with RFIDTagSensors.*
- class **gazebo::sensors::Sensor**

  *Base class for sensors.*
- class **SensorFactor**

  *The sensor factory; the class is just for namespacing purposes.*
- class **gazebo::sensors::SensorFactory**
- class **gazebo::sensors::SensorManager**

  *Class to manage and update all sensors.*
- class **gazebo::sensors::SonarSensor**

***Sensor*** *(p. 837) with sonar cone.*

- class **gazebo::sensors::WirelessReceiver**

  ***Sensor*** *(p. 837) class for receiving wireless signals.*

- class **gazebo::sensors::WirelessTransceiver**

  ***Sensor*** *(p. 837) class for receiving wireless signals.*

- class **gazebo::sensors::WirelessTransmitter**

  *Transmitter to send wireless signals.*

## Macros

- #define **GZ_REGISTER_STATIC_SENSOR**(name, classname)

  *Static sensor registration macro.*

## Functions

- std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(2.0)

  *Deprecated.*

- std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  *Create a sensor using SDF.*

- void **gazebo::sensors::disable** ()

  *Disable sensors.*

- void **gazebo::sensors::enable** ()

  *Enable sensors.*

- bool **gazebo::sensors::fini** ()

  *shutdown the sensor generation loop.*

- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)

  *Get a sensor using by name.*

- bool **gazebo::sensors::init** ()

  *initialize the sensor generation loop.*

- bool **gazebo::sensors::load** ()

  *Load the sensor library.*

- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)

  *Remove a sensor by name.*

- bool **gazebo::sensors::remove_sensors** ()

  *Remove all sensors.*

- void **gazebo::sensors::run_once** (bool _force=false)

  *Run the sensor generation one step.*

- void **gazebo::sensors::run_threads** ()

  *Run sensors in a threads. This is a non-blocking call.*

- void **gazebo::sensors::stop** ()

  *Stop the sensor generation loop.*

### 8.10.1 Detailed Description

A set of sensor classes, functions, and definitions. GPU based laser sensor.

Depth camera sensor This sensor is used for simulating standard monocular cameras

This sensor cast rays into the world, tests for intersections, and reports the range to the nearest object. It is used by ranging sensor models (e.g., sonars and scanning laser range finders).

### 8.10.2 Macro Definition Documentation

#### 8.10.2.1 #define GZ_REGISTER_STATIC_SENSOR( *name, classname* )

**Value:**

```
Sensor *New##classname() \
  { \
    return new gazebo::sensors::classname(); \
  } \
  void Register##classname() \
  {\
    SensorFactory::RegisterSensor(name, New##classname);\
  }
```

Static sensor registration macro.

Use this macro to register sensors with the server.

**Parameters**

| | |
|---|---|
| *name* | Sensor type name, as it appears in the world file. |
| *classname* | C++ class name for the sensor. |

### 8.10.3 Function Documentation

#### 8.10.3.1 std::string gazebo::sensors::create_sensor ( sdf::ElementPtr *_elem,* const std::string & *_worldName,* const std::string & *_parentName* )

Deprecated.

#### 8.10.3.2 std::string gazebo::sensors::create_sensor ( sdf::ElementPtr *_elem,* const std::string & *_worldName,* const std::string & *_parentName,* uint32_t *_parentId* )

Create a sensor using SDF.

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | The SDF element that describes the sensor. |
| in | *_worldName* | Name of the world in which to create the sensor. |
| in | *_parentName* | The fully scoped parent name (model::link). |

**Returns**

The name of the new sensor.

#### 8.10.3.3 void gazebo::sensors::disable ( )

Disable sensors.

#### 8.10.3.4 void gazebo::sensors::enable ( )

Enable sensors.

#### 8.10.3.5 bool gazebo::sensors::fini ( )

shutdown the sensor generation loop.

**Returns**

True if successfully finalized, false if not

#### 8.10.3.6 SensorPtr gazebo::sensors::get_sensor ( const std::string & *_name* )

Get a sensor using by name.

The given name should have: world_name::model_name::link_name::sensor_name

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the sensor. This name should be fully scoped. This means _name = world_name::model_name::link_name::sensor_name. You may use the unscoped sensor name if that name is unique within the entire simulation. If the name is not unique a NULL pointer is returned. |

**Returns**

Pointer to the sensor, NULL if the sensor could not be found.

#### 8.10.3.7 bool gazebo::sensors::init ( )

initialize the sensor generation loop.

**Returns**

True if successfully initialized, false if not

#### 8.10.3.8 bool gazebo::sensors::load ( )

Load the sensor library.

**Returns**

True if successfully loaded, false if not.

#### 8.10.3.9 void gazebo::sensors::remove_sensor ( const std::string & _sensorName )

Remove a sensor by name.

**Parameters**

| | | |
|---|---|---|
| in | *_sensorName* | Name of sensor to remove |

#### 8.10.3.10 bool gazebo::sensors::remove_sensors ( )

Remove all sensors.

**Returns**

True if all successfully removed, false if not

#### 8.10.3.11 void gazebo::sensors::run_once ( bool _force = false )

Run the sensor generation one step.

**Parameters**

| | |
|---|---|
| *_force,:* | If true, all sensors are forced to update. Otherwise a sensor will update based on it's Hz rate. |

#### 8.10.3.12 void gazebo::sensors::run_threads ( )

Run sensors in a threads. This is a non-blocking call.

#### 8.10.3.13 void gazebo::sensors::stop ( )

Stop the sensor generation loop.

## 8.11 Transport

Handles transportation of messages.

### Files

- file **TransportTypes.hh**

  *Forward declarations for transport.*

### Classes

- class **gazebo::transport::CallbackHelper**

  *A helper class to handle callbacks when messages arrive.*
- class **gazebo::transport::CallbackHelperT< M >**

  *Callback helper Template.*
- class **gazebo::transport::Connection**

  *Single TCP/IP connection manager.*
- class **gazebo::transport::ConnectionManager**

  *Manager of connections.*
- class **gazebo::transport::IOManager**

  *Manages boost::asio IO.*
- class **gazebo::transport::Node**

  *A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*
- class **gazebo::transport::Publication**

  *A publication for a topic.*
- class **gazebo::transport::PublicationTransport**

  *transport/transport.hh*
- class **gazebo::transport::Publisher**

  *A publisher of messages on a topic.*
- class **gazebo::transport::RawCallbackHelper**

  *Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.*
- class **gazebo::transport::SubscribeOptions**

  *Options for a subscription.*
- class **gazebo::transport::Subscriber**

  *A subscriber to a topic.*
- class **gazebo::transport::SubscriptionTransport**

  *transport/transport.hh*
- class **gazebo::transport::TopicManager**

  *Manages topics and their subscriptions.*

### Typedefs

- typedef boost::shared_ptr
  < CallbackHelper > **gazebo::transport::CallbackHelperPtr**

  *boost shared pointer to **transport::CallbackHelper** (p. 180)*

## Functions

- void **gazebo::transport::clear_buffers** ()

  *Clear any remaining communication buffers.*
- void **gazebo::transport::fini** ()

  *Cleanup the transport component.*
- bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)

  *Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.*
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)

  *Return all the namespace (world names) on the master.*
- std::map< std::string,
  std::list< std::string > > **gazebo::transport::getAdvertisedTopics** ()

  *Get a list of all the topics and their message types.*
- std::list< std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)

  *Get a list of all the unique advertised topic names.*
- bool **gazebo::transport::getMinimalComms** ()

  *Get whether minimal comms has been enabled.*
- std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)

  *Get the message typename that is published on the given topic.*
- bool **gazebo::transport::init** (const std::string &_master_host="", unsigned int _master_port=0)

  *Initialize the transport system.*
- bool **gazebo::transport::is_stopped** ()

  *Is the transport system stopped?*
- void **gazebo::transport::pause_incoming** (bool _pause)

  *Pause or unpause incoming messages.*
- template<typename M >
  void **gazebo::transport::publish** (const std::string &_topic, const google::protobuf::Message &_message)

  *A convenience function for a one-time publication of a message.*
- boost::shared_ptr< msgs::Response > **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  *Send a request and receive a response.*
- void **gazebo::transport::requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  *Send a request and don't wait for a response.*
- void **gazebo::transport::requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")

  *Send a request and don't wait for a response.*
- void **gazebo::transport::run** ()

  *Run the transport component.*
- void **gazebo::transport::setMinimalComms** (bool _enabled)

  *Set whether minimal comms should be used.*
- void **gazebo::transport::stop** ()

  *Stop the transport component from running.*

### 8.11.1 Detailed Description

Handles transportation of messages.

**Remarks**

Environment Variables:

- GAZEBO_IP_WHITE_LIST: Comma separated list of valid IPs. Leave this empty to accept connections from all addresses.
- GAZEBO_IP: IP address to export. This will override the default IP lookup.
- GAZEBO_HOSTNAME: Hostame to export. Setting this will override both GAZEBO_IP and the default IP lookup.

### 8.11.2 Typedef Documentation

#### 8.11.2.1 typedef boost::shared_ptr<CallbackHelper> gazebo::transport::CallbackHelperPtr

boost shared pointer to **transport::CallbackHelper** (p. 180)

### 8.11.3 Function Documentation

#### 8.11.3.1 void gazebo::transport::clear_buffers ( )

Clear any remaining communication buffers.

#### 8.11.3.2 void gazebo::transport::fini ( )

Cleanup the transport component.

#### 8.11.3.3 bool gazebo::transport::get_master_uri ( std::string & *_master_host,* unsigned int & *_master_port* )

Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.

**Parameters**

| | | |
|---|---|---|
| out | *_master_host* | The hostname of the master is set to this param |
| out | *_master_port* | The port of the master is set to this param |

**Returns**

true if GAZEBO_MASTER_URI was successfully parsed; false otherwise (in which case output params are not set)

#### 8.11.3.4 void gazebo::transport::get_topic_namespaces ( std::list< std::string > & *_namespaces* )

Return all the namespace (world names) on the master.

**Parameters**

| | | |
|---|---|---|
| out | *_namespaces* | The list of namespace will be written here |

#### 8.11.3.5 std::map<std::string, std::list<std::string> > gazebo::transport::getAdvertisedTopics ( )

Get a list of all the topics and their message types.

**Returns**

A map where keys are message types, and values are a list of topic names.

#### 8.11.3.6 std::list<std::string> gazebo::transport::getAdvertisedTopics ( const std::string & *_msgType* )

Get a list of all the unique advertised topic names.

**Parameters**

| | | |
|---|---|---|
| in | *_msgType* | Type of message to filter the result on. If empty, then a list of all the topics is returned. |

**Returns**

A list of the advertised topics that publish messages of the type specified by _msgType.

#### 8.11.3.7 bool gazebo::transport::getMinimalComms ( )

Get whether minimal comms has been enabled.

**Returns**

True if minimal comms is enabled.

#### 8.11.3.8 std::string gazebo::transport::getTopicMsgType ( const std::string & *_topicName* )

Get the message typename that is published on the given topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topicName* | Name of the topic to query. |

**Returns**

The message type, or empty string if the topic is not valid.

#### 8.11.3.9 bool gazebo::transport::init ( const std::string & *_master_host* = " ", unsigned int *_master_port* = 0 )

Initialize the transport system.

**Parameters**

| | | |
|---|---|---|
| in | *_master_host* | The hostname or IP of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var. |
| in | *_master_port* | The port of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var. |

**Returns**

true if initialization succeeded; false otherwise

#### 8.11.3.10 bool gazebo::transport::is_stopped ( )

Is the transport system stopped?

**Returns**

true if the transport system is stopped; false otherwise

#### 8.11.3.11 void gazebo::transport::pause_incoming ( bool *_pause* )

Pause or unpause incoming messages.

When paused, messages are queued for later delivery

**Parameters**

| | | |
|---|---|---|
| in | *_pause* | If true, pause; otherwise unpause |

#### 8.11.3.12 template<typename M > void gazebo::transport::publish ( const std::string & *_topic,* const google::protobuf::Message & *_message* )

A convenience function for a one-time publication of a message.

This is inefficient, compared to **Node::Advertise** (p. 674) followed by **Publisher::Publish** (p. 759). This function should only be used when sending a message very infrequently.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to advertise |
| in | *_message* | Message to be published |

#### 8.11.3.13 boost::shared_ptr<msgs::Response> gazebo::transport::request ( const std::string & *_worldName,* const std::string & *_request,* const std::string & *_data* = " " )

Send a request and receive a response.

This call will block until a response is received.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | The name of the world to which the request should be sent |
| in | *_request* | The type request. |
| in | *_data* | Optional data string. |

**Returns**

The response to the request. Can be empty.

#### 8.11.3.14 void gazebo::transport::requestNoReply ( const std::string & _worldName, const std::string & _request, const std::string & _data = " " )

Send a request and don't wait for a response.

This is non-blocking.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | The name of the world to which the request should be sent. |
| in | *_request* | The type request. |
| in | *_data* | Optional data string. |

#### 8.11.3.15 void gazebo::transport::requestNoReply ( NodePtr _node, const std::string & _request, const std::string & _data = " " )

Send a request and don't wait for a response.

This is non-blocking.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | Pointer to a node that provides communication. |
| in | *_request* | The type request. |
| in | *_data* | Optional data string. |

#### 8.11.3.16 void gazebo::transport::run ( )

Run the transport component.

Creates a thread to handle message passing. This call will block until the master can be contacted or until a retry limit is reached

#### 8.11.3.17 void gazebo::transport::setMinimalComms ( bool _enabled )

Set whether minimal comms should be used.

This will be used to reduce network traffic.

#### 8.11.3.18 void gazebo::transport::stop ( )

Stop the transport component from running.

## 8.12 Utility

### Files

- file **UtilTypes.hh**

### Classes

- class **gazebo::util::DiagnosticManager**

  *A diagnostic manager class.*
- class **gazebo::util::DiagnosticTimer**

  *A timer designed for diagnostics.*
- class **gazebo::util::OpenAL**

  *3D audio setup and playback.*
- class **gazebo::util::OpenALSink**

  ***OpenAL*** *(*p. *695) Listener.*
- class **gazebo::util::OpenALSource**

  ***OpenAL*** *(*p. *695) Source.*

### Macros

- #define **DIAG_TIMER_LAP**(_name, _prefix) ((void)0)
- #define **DIAG_TIMER_START**(_name) ((void) 0)
- #define **DIAG_TIMER_STOP**(_name) ((void) 0)

### 8.12.1 Detailed Description

### 8.12.2 Macro Definition Documentation

#### 8.12.2.1 #define DIAG_TIMER_LAP( *_name, _prefix* ) ((void)0)

#### 8.12.2.2 #define DIAG_TIMER_START( *_name* ) ((void) 0)

#### 8.12.2.3 #define DIAG_TIMER_STOP( *_name* ) ((void) 0)

# Chapter 9

# Namespace Documentation

## 9.1 boost Namespace Reference

## 9.2 gazebo Namespace Reference

Forward declarations for the common classes.

### Namespaces

- namespace **common**

  *Common namespace.*
- namespace **event**

  ***Event*** *(p. 390) namespace.*
- namespace **math**

  *Math namespace.*
- namespace **msgs**

  *Messages namespace.*
- namespace **physics**

  *namespace for physics*
- namespace **rendering**

  *Rendering namespace.*
- namespace **sensors**

  *Sensors namespace.*
- namespace **transport**
- namespace **util**

### Classes

- class **Master**

  *A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.*
- class **ModelPlugin**

  *A plugin with access to* ***physics::Model*** *(p. 624).*

- class **PluginT**

  *A class which all plugins must inherit from.*
- class **SensorPlugin**

  *A plugin with access to physics::Sensor.*
- class **Server**
- class **SystemPlugin**

  *A plugin loaded within the gzserver on startup.*
- class **VisualPlugin**

  *A plugin loaded within the gzserver on startup.*
- class **WorldPlugin**

  *A plugin with access to **physics::World*** (p. *1157*).

### Typedefs

- typedef boost::shared_ptr < GUIPlugin > **GUIPluginPtr**
- typedef boost::shared_ptr < **ModelPlugin** > **ModelPluginPtr**
- typedef boost::shared_ptr < **SensorPlugin** > **SensorPluginPtr**
- typedef boost::shared_ptr < **SystemPlugin** > **SystemPluginPtr**
- typedef boost::shared_ptr < **VisualPlugin** > **VisualPluginPtr**
- typedef boost::shared_ptr < **WorldPlugin** > **WorldPluginPtr**

### Enumerations

- enum **PluginType** { **WORLD_PLUGIN**, **MODEL_PLUGIN**, **SENSOR_PLUGIN**, **SYSTEM_PLUGIN**, **VISUAL_PLUGIN** }

  *Used to specify the type of plugin.*

### Functions

- void **add_plugin** (const std::string &_filename)
- std::string **find_file** (const std::string &_file)

  *Find a file in the gazebo search paths.*
- void **fini** ()
- bool **init** ()
- bool **load** (int _argc=0, char **_argv=0)
- void **print_version** ()
- void **run** ()
- void **stop** ()

#### 9.2.1 Detailed Description

Forward declarations for the common classes. Forward declarations for the util classes.

### 9.2.2 Typedef Documentation

#### 9.2.2.1 typedef boost::shared_ptr<GUIPlugin> gazebo::GUIPluginPtr

#### 9.2.2.2 typedef boost::shared_ptr<ModelPlugin> gazebo::ModelPluginPtr

#### 9.2.2.3 typedef boost::shared_ptr<SensorPlugin> gazebo::SensorPluginPtr

#### 9.2.2.4 typedef boost::shared_ptr<SystemPlugin> gazebo::SystemPluginPtr

#### 9.2.2.5 typedef boost::shared_ptr<VisualPlugin> gazebo::VisualPluginPtr

#### 9.2.2.6 typedef boost::shared_ptr<WorldPlugin> gazebo::WorldPluginPtr

### 9.2.3 Function Documentation

#### 9.2.3.1 void gazebo::add_plugin ( const std::string & *_filename* )

#### 9.2.3.2 std::string gazebo::find_file ( const std::string & *_file* )

Find a file in the gazebo search paths.

#### 9.2.3.3 void gazebo::fini ( )

#### 9.2.3.4 bool gazebo::init ( )

#### 9.2.3.5 bool gazebo::load ( int *_argc* = 0, char ** *_argv* = 0 )

#### 9.2.3.6 void gazebo::print_version ( )

#### 9.2.3.7 void gazebo::run ( )

#### 9.2.3.8 void gazebo::stop ( )

## 9.3 gazebo::common Namespace Reference

Common namespace.

### Classes

- class **Animation**

  *Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.*
- class **AssertionInternalError**

  *Class for generating Exceptions which come from gazebo assertions.*
- class **AudioDecoder**

  *An audio decoder based on FFMPEG.*
- class **BVHLoader**

  *Handles loading BVH animation files.*
- class **ColladaLoader**

*Class used to load Collada mesh files.*

- class **Color**

  *Defines a color.*
- class **Console**

  *Message, error, warning functionality.*
- class **Exception**

  *Class for generating exceptions.*
- class **Image**

  *Encapsulates an image.*
- class **InternalError**

  *Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.*
- class **KeyEvent**

  *Generic description of a keyboard event.*
- class **KeyFrame**

  *A key frame in an animation.*
- class **Material**

  *Encapsulates description of a material.*
- class **Mesh**

  *A 3D mesh.*
- class **MeshCSG**

  *Creates CSG meshes.*
- class **MeshLoader**

  *Base class for loading meshes.*
- class **MeshManager**

  *Maintains and manages all meshes.*
- class **ModelDatabase**

  *Connects to model database, and has utility functions to find models.*
- class **MouseEvent**

  *Generic description of a mouse event.*
- class **NodeAnimation**

  *Node animation.*
- struct **NodeAssignment**

  *Vertex to node weighted assignement for skeleton animation visualization.*
- class **NodeTransform**

  ***NodeTransform*** *(*p. *684)* ***Skeleton.hh*** *(*p. *1367) common/common.hh*
- class **NumericAnimation**

  *A numeric animation.*
- class **NumericKeyFrame**

  *A keyframe for a* ***NumericAnimation*** *(*p. *692).*
- class **ParamT**
- class **PID**

  *Generic* ***PID*** *(*p. *722) controller class.*
- class **PoseAnimation**

  *A pose animation.*
- class **PoseKeyFrame**

  *A keyframe for a* ***PoseAnimation*** *(*p. *743).*
- class **Skeleton**

*A skeleton.*

- class **SkeletonAnimation**

  ***Skeleton*** *(*p. *953) animation.*

- class **SkeletonNode**

  *A skeleton node.*

- class **SphericalCoordinates**

  *Convert spherical coordinates for planetary surfaces.*

- class **STLLoader**

  *Class used to load STL mesh files.*

- class **SubMesh**

  *A child mesh.*

- class **SystemPaths**

  *Functions to handle getting system paths, keeps track of:*

- class **Time**

  *A* ***Time*** *(*p. *1031) class, can be used to hold wall- or sim-time.*

- class **Timer**

  *A timer class, used to time things in real world walltime.*

- class **UpdateInfo**

  *Information for use in an update event.*

- class **Video**

  *Handle video encoding and decoding using libavcodec.*

## Typedefs

- typedef boost::shared_ptr
  < **Animation** > **AnimationPtr**
- typedef boost::shared_ptr
  < DiagnosticTimer > **DiagnosticTimerPtr**
- typedef std::map< unsigned int,
  **SkeletonNode** * > **NodeMap**
- typedef std::map< unsigned int,
  **SkeletonNode** * >::iterator **NodeMapIter**
- typedef boost::shared_ptr
  < **NumericAnimation** > **NumericAnimationPtr**
- typedef std::vector
  < common::Param * > **Param_V**
- typedef boost::shared_ptr
  < **PoseAnimation** > **PoseAnimationPtr**
- typedef std::map< double,
  std::vector< **NodeTransform** > > **RawNodeAnim**
- typedef std::vector
  < std::vector< std::pair
  < std::string, double > > > **RawNodeWeights**
- typedef std::map< std::string,
  **RawNodeAnim** > **RawSkeletonAnim**
- typedef boost::shared_ptr
  < **SphericalCoordinates** > **SphericalCoordinatesPtr**
- typedef std::map< std::string,
  std::string > **StrStr_M**

### Functions

- void **add_search_path_suffix** (const std::string &_suffix)

  *add path sufix to **common::SystemPaths*** (p. *1025*)
- std::string **find_file** (const std::string &_file)

  *search for file in **common::SystemPaths*** (p. *1025*)
- std::string **find_file** (const std::string &_file, bool _searchLocalPath)

  *search for file in **common::SystemPaths*** (p. *1025*)
- std::string **find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths*** (p. *1025*)
- template<typename T >
  std::string **get_sha1** (const T &_buffer)

  *Compute the SHA1 hash of an array of bytes.*
- void **load** ()

  *Load the common library.*

### Variables

- static std::string **PixelFormatNames** [ ]

  *String names for the pixel formats.*
- static const double **SpeedOfLight** = 299792458

  *Speed of light.*

#### 9.3.1 Detailed Description

Common namespace.

#### 9.3.2 Typedef Documentation

##### 9.3.2.1 typedef boost::shared_ptr<Animation> gazebo::common::AnimationPtr

##### 9.3.2.2 typedef boost::shared_ptr<DiagnosticTimer> gazebo::common::DiagnosticTimerPtr

##### 9.3.2.3 typedef std::map<unsigned int, SkeletonNode*> gazebo::common::NodeMap

##### 9.3.2.4 typedef std::map<unsigned int, SkeletonNode*>::iterator gazebo::common::NodeMapIter

##### 9.3.2.5 typedef boost::shared_ptr<NumericAnimation> gazebo::common::NumericAnimationPtr

##### 9.3.2.6 typedef std::vector<common::Param*> gazebo::common::Param_V

##### 9.3.2.7 typedef boost::shared_ptr<PoseAnimation> gazebo::common::PoseAnimationPtr

##### 9.3.2.8 typedef std::map<double, std::vector<NodeTransform> > gazebo::common::RawNodeAnim

##### 9.3.2.9 typedef std::vector<std::vector<std::pair<std::string, double> > > gazebo::common::RawNodeWeights

##### 9.3.2.10 typedef std::map<std::string, RawNodeAnim> gazebo::common::RawSkeletonAnim

#### 9.3.2.11 typedef boost::shared_ptr<SphericalCoordinates> gazebo::common::SphericalCoordinatesPtr

#### 9.3.2.12 typedef std::map<std::string, std::string> gazebo::common::StrStr_M

### 9.3.3 Variable Documentation

#### 9.3.3.1 const double gazebo::common::SpeedOfLight = 299792458 [static]

Speed of light.

## 9.4 gazebo::event Namespace Reference

**Event** (p. 390) namespace.

### Classes

- class **Connection**

  *A class that encapsulates a connection.*
- class **Event**

  *Base class for all events.*
- class **Events**

  *An **Event** (p. 390) class to get notifications for simulator events.*
- class **EventT**

  *A class for event processing.*

### Typedefs

- typedef std::vector < **ConnectionPtr** > **Connection_V**
- typedef boost::shared_ptr < **Connection** > **ConnectionPtr**

### 9.4.1 Detailed Description

**Event** (p. 390) namespace.

### 9.4.2 Typedef Documentation

#### 9.4.2.1 typedef std::vector<ConnectionPtr> gazebo::event::Connection_V

#### 9.4.2.2 typedef boost::shared_ptr<Connection> gazebo::event::ConnectionPtr

## 9.5 gazebo::math Namespace Reference

Math namespace.

## Classes

- class **Angle**

  *An angle and related functions.*
- class **Box**

  *Mathematical representation of a box and related functions.*
- class **Matrix3**

  *A 3x3 matrix class.*
- class **Matrix4**

  *A 3x3 matrix class.*
- class **Plane**

  *A plane and related functions.*
- class **Pose**

  *Encapsulates a position and rotation in three space.*
- class **Quaternion**

  *A quaternion class.*
- class **Rand**

  *Random number generator class.*
- class **RotationSpline**

  ***Spline*** *(p. 993) for rotations.*
- class **Spline**

  *Splines.*
- class **Vector2d**

  *Generic double x, y vector.*
- class **Vector2i**

  *Generic integer x, y vector.*
- class **Vector3**

  *The* ***Vector3*** *(p. 1091) class represents the generic vector containing 3 elements.*
- class **Vector4**

  *double Generic x, y, z, w vector*

## Typedefs

- typedef boost::mt19937 **GeneratorType**
- typedef boost::normal_distribution < double > **NormalRealDist**
- typedef boost::variate_generator < **GeneratorType** &, **NormalRealDist** > **NRealGen**
- typedef boost::variate_generator < **GeneratorType** &, **UniformIntDist** > **UIntGen**
- typedef boost::uniform_int< int > **UniformIntDist**
- typedef boost::uniform_real < double > **UniformRealDist**

- typedef
  boost::variate_generator
  < **GeneratorType**
  &, **UniformRealDist** > **URealGen**

## Functions

- template<typename T >
  T **clamp** (T _v, T _min, T _max)

  *Simple clamping function.*
- template<typename T >
  bool **equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

  *check if two values are equal, within a tolerance*
- float **fixnan** (float _v)

  *Fix a nan value.*
- double **fixnan** (double _v)

  *Fix a nan value.*
- bool **isnan** (float _v)

  *check if a float is NaN*
- bool **isnan** (double _v)

  *check if a double is NaN*
- bool **isPowerOfTwo** (unsigned int _x)

  *is this a power of 2?*
- template<typename T >
  T **max** (const std::vector< T > &_values)

  *get the maximum value of vector of values*
- template<typename T >
  T **mean** (const std::vector< T > &_values)

  *get mean of vector of values*
- template<typename T >
  T **min** (const std::vector< T > &_values)

  *get the minimum value of vector of values*
- double **parseFloat** (const std::string &_input)

  *parse string into float*
- int **parseInt** (const std::string &_input)

  *parse string into an integer*
- template<typename T >
  T **precision** (const T &_a, const unsigned int &_precision)

  *get value at a specified precision*
- template<typename T >
  T **variance** (const std::vector< T > &_values)

  *get variance of vector of values*

## Variables

- static const double **NAN_D** = std::numeric_limits<double>::quiet_NaN()

  *Returns the representation of a quiet not a number (NAN)*
- static const int **NAN_I** = std::numeric_limits<int>::quiet_NaN()

  *Returns the representation of a quiet not a number (NAN)*

### 9.5.1 Detailed Description

Math namespace.

### 9.5.2 Typedef Documentation

#### 9.5.2.1 typedef boost::mt19937 gazebo::math::GeneratorType

#### 9.5.2.2 typedef boost::normal_distribution<double> gazebo::math::NormalRealDist

#### 9.5.2.3 typedef boost::variate_generator<GeneratorType&, NormalRealDist > gazebo::math::NRealGen

#### 9.5.2.4 typedef boost::variate_generator<GeneratorType&, UniformIntDist > gazebo::math::UIntGen

#### 9.5.2.5 typedef boost::uniform_int<int> gazebo::math::UniformIntDist

#### 9.5.2.6 typedef boost::uniform_real<double> gazebo::math::UniformRealDist

#### 9.5.2.7 typedef boost::variate_generator<GeneratorType&, UniformRealDist > gazebo::math::URealGen

## 9.6 gazebo::msgs Namespace Reference

Messages namespace.

### Classes

- class **MsgFactory**

  *A factory that generates protobuf message based on a string type.*

### Typedefs

- typedef boost::shared_ptr
  < google::protobuf::Message >(* **MsgFactoryFn** )()

### Functions

- msgs::Vector3d **Convert** (const **math::Vector3** &_v)

  *Convert a **math::Vector3** (p. 1091) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const **math::Quaternion** &_q)

  *Convert a **math::Quaternion** (p. 761) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const **math::Pose** &_p)

  *Convert a **math::Pose** (p. 734) to a msgs::Pose.*
- msgs::Color **Convert** (const **common::Color** &_c)

  *Convert a **common::Color** (p. 233) to a msgs::Color.*
- msgs::Time **Convert** (const **common::Time** &_t)

  *Convert a **common::Time** (p. 1031) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const **math::Plane** &_p)

*Convert a **math::Plane** (p. 726) to a msgs::PlaneGeom.*

- **math::Vector3 Convert** (const msgs::Vector3d &_v)

  *Convert a msgs::Vector3d to a math::Vector.*
- **math::Quaternion Convert** (const msgs::Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion** (p. 761).*
- **math::Pose Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 734).*
- **common::Color Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 233).*
- **common::Time Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 1031).*
- **math::Plane Convert** (const msgs::PlaneGeom &_p)

  *Convert a msgs::PlaneGeom to a common::Plane.*
- msgs::Request * **CreateRequest** (const std::string &_request, const std::string &_data="")

  *Create a request message.*
- msgs::Fog **FogFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Fog from a fog SDF element.*
- msgs::Geometry **GeometryFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Geometry from a geometry SDF element.*
- msgs::Header * **GetHeader** (google::protobuf::Message &_message)

  *Get the header from a protobuf message.*
- msgs::GUI **GUIFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::GUI from a GUI SDF element.*
- void **Init** (google::protobuf::Message &_message, const std::string &_id="")

  *Initialize a message.*
- msgs::Light **LightFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Light from a light SDF element.*
- msgs::MeshGeom **MeshFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::MeshGeom from a mesh SDF element.*
- msgs::Scene **SceneFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Scene from a scene SDF element.*
- void **Set** (**common::Image** &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 474).*
- void **Set** (msgs::Image *_msg, const **common::Image** &_i)

  *Set a msgs::Image from a **common::Image** (p. 474).*
- void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 1091).*
- void **Set** (msgs::Vector2d *_pt, const **math::Vector2d** &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 1091).*
- void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 761).*
- void **Set** (msgs::Pose *_p, const **math::Pose** &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 734).*
- void **Set** (msgs::Color *_c, const **common::Color** &_v)

  *Set a msgs::Color from a **common::Color** (p. 233).*
- void **Set** (msgs::Time *_t, const **common::Time** &_v)

  *Set a msgs::Time from a **common::Time** (p. 1031).*

- void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)

  *Set a msgs::Plane from a **math::Plane*** (p. *726*).
- void **Stamp** (msgs::Header *_header)

  *Time stamp a header.*
- void **Stamp** (msgs::Time *_time)

  *Set the time in a time message.*
- msgs::TrackVisual **TrackVisualFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::TrackVisual from a track visual SDF element.*
- msgs::Visual **VisualFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Visual from a visual SDF element.*

### 9.6.1 Detailed Description

Messages namespace.

### 9.6.2 Typedef Documentation

#### 9.6.2.1 typedef boost::shared_ptr<google::protobuf::Message>(* gazebo::msgs::MsgFactoryFn)()

## 9.7 gazebo::physics Namespace Reference

namespace for physics

### Classes

- class **Actor**

  ***Actor*** (p. *131*) *class enables GPU based mesh model / skeleton scriptable animation.*
- class **BallJoint**

  ***Base*** (p. *159*) *class for a ball joint.*
- class **Base**

  ***Base*** (p. *159*) *class for most physics classes.*
- class **BoxShape**

  *Box geometry primitive.*
- class **Collision**

  ***Base*** (p. *159*) *class for all collision entities.*
- class **CollisionState**

  *Store state information of a **physics::Collision*** (p. *220*) *object.*
- class **Contact**

  *A contact between two collisions.*
- class **ContactManager**

  *Aggregates all the contact information generated by the collision detection engine.*
- class **ContactPublisher**

  *A custom contact publisher created for each contact filter in the **Contact*** (p. *260*) *Manager.*
- class **CylinderShape**

  *Cylinder collision.*
- class **DARTBallJoint**

*An **DARTBallJoint** (p. 279).*

- class **DARTBoxShape**

  *DART Box shape.*
- class **DARTCollision**

  ***Base** (p. 159) class for all DART collisions.*
- class **DARTCylinderShape**

  *DART cylinder shape.*
- class **DARTHeightmapShape**

  *DART Height map collision.*
- class **DARTHinge2Joint**

  *A two axis hinge joint.*
- class **DARTHingeJoint**

  *A single axis hinge joint.*
- class **DARTJoint**

  *DART joint interface.*
- class **DARTLink**

  *DART **Link** (p. 542) class.*
- class **DARTMeshShape**

  *Triangle mesh collision.*
- class **DARTModel**

  *DART model class.*
- class **DARTMultiRayShape**

  *DART specific version of **MultiRayShape** (p. 664).*
- class **DARTPhysics**

  *DART physics engine.*
- class **DARTPlaneShape**

  *An DART Plane shape.*
- class **DARTRayShape**

  *Ray collision.*
- class **DARTScrewJoint**

  *A screw joint.*
- class **DARTSliderJoint**

  *A slider joint.*
- class **DARTSphereShape**

  *A DART sphere shape.*
- class **DARTTypes**

  *A set of functions for converting between the math types used by gazebo and dart.*
- class **DARTUniversalJoint**

  *A universal joint.*
- class **Entity**

  ***Base** (p. 159) class for all physics objects in Gazebo.*
- class **Gripper**

  *A gripper abstraction.*
- class **HeightmapShape**

  ***HeightmapShape** (p. 465) collision shape builds a heightmap from an image.*
- class **Hinge2Joint**

  *A two axis hinge joint.*

- class **HingeJoint**

  *A single axis hinge joint.*
- class **Inertial**

  *A class for inertial information about a link.*
- class **Joint**

  ***Base*** *(*p. *159) class for all joints.*
- class **JointController**

  *A class for manipulating* ***physics::Joint*** *(*p. *496).*
- class **JointState**

  *keeps track of state of a* ***physics::Joint*** *(*p. *496)*
- class **JointWrench**

  *Wrench information from a joint.*
- class **Link**

  ***Link*** *(*p. *542) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*
- class **LinkState**

  *Store state information of a* ***physics::Link*** *(*p. *542) object.*
- class **MapShape**

  *Creates box extrusions based on an image.*
- class **MeshShape**

  *Triangle mesh collision shape.*
- class **Model**

  *A model is a collection of links, joints, and plugins.*
- class **ModelState**

  *Store state information of a* ***physics::Model*** *(*p. *624) object.*
- class **MultiRayShape**

  *Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.*
- class **PhysicsEngine**

  ***Base*** *(*p. *159) class for a physics engine.*
- class **PhysicsFactory**

  *The physics factory instantiates different physics engines.*
- class **PlaneShape**

  ***Collision*** *(*p. *220) for an infinite plane.*
- class **RayShape**

  ***Base*** *(*p. *159) class for Ray collision geometry.*
- class **Road**

  *for building a* ***Road*** *(*p. *804) from SDF*
- class **ScrewJoint**

  *A screw joint, which has both prismatic and rotational DOFs.*
- class **Shape**

  ***Base*** *(*p. *159) class for all shapes.*
- class **SimbodyBallJoint**

  ***SimbodyBallJoint*** *(*p. *865) class models a ball joint in Simbody.*
- class **SimbodyBoxShape**

  *Simbody box collision.*
- class **SimbodyCollision**

  *Simbody collisions.*

- class **SimbodyCylinderShape**

  *Cylinder collision.*
- class **SimbodyHeightmapShape**

  *Height map collision.*
- class **SimbodyHinge2Joint**

  *A two axis hinge joint.*
- class **SimbodyHingeJoint**

  *A single axis hinge joint.*
- class **SimbodyJoint**

  ***Base*** *(*p. *159) class for all joints.*
- class **SimbodyLink**

  *Simbody* ***Link*** *(*p. *542) class.*
- class **SimbodyMeshShape**

  *Triangle mesh collision.*
- class **SimbodyModel**

  *A model is a collection of links, joints, and plugins.*
- class **SimbodyMultiRayShape**

  *Simbody specific version of* ***MultiRayShape*** *(*p. *664).*
- class **SimbodyPhysics**

  *Simbody physics engine.*
- class **SimbodyPlaneShape**

  *Simbody collision for an infinite plane.*
- class **SimbodyRayShape**

  *Ray shape for simbody.*
- class **SimbodyScrewJoint**

  *A screw joint.*
- class **SimbodySliderJoint**

  *A slider joint.*
- class **SimbodySphereShape**

  *Simbody sphere collision.*
- class **SimbodyUniversalJoint**

  *A simbody universal joint class.*
- class **SliderJoint**

  *A slider joint.*
- class **SphereShape**

  *Sphere collision shape.*
- class **State**

  ***State*** *(*p. *998) of an entity.*
- class **SurfaceParams**

  ***SurfaceParams*** *(*p. *1020) defines various Surface contact parameters.*
- struct **TrajectoryInfo**
- class **UniversalJoint**

  *A universal joint.*
- class **World**

  *The world provides access to all other object within a simulated environment.*
- class **WorldState**

  *Store state information of a* ***physics::World*** *(*p. *1157) object.*

## Typedefs

- typedef std::vector< **ActorPtr** > **Actor_V**
- typedef boost::shared_ptr< **Actor** > **ActorPtr**
- typedef std::vector< **BasePtr** > **Base_V**
- typedef boost::shared_ptr< **Base** > **BasePtr**
- typedef boost::shared_ptr < **BoxShape** > **BoxShapePtr**
- typedef std::vector< **CollisionPtr** > **Collision_V**
- typedef boost::shared_ptr < **Collision** > **CollisionPtr**
- typedef boost::shared_ptr < **Contact** > **ContactPtr**
- typedef boost::shared_ptr < **CylinderShape** > **CylinderShapePtr**
- typedef boost::shared_ptr < **DARTCollision** > **DARTCollisionPtr**
- typedef boost::shared_ptr < **DARTJoint** > **DARTJointPtr**
- typedef boost::shared_ptr < **DARTLink** > **DARTLinkPtr**
- typedef boost::shared_ptr < **DARTModel** > **DARTModelPtr**
- typedef boost::shared_ptr < **DARTPhysics** > **DARTPhysicsPtr**
- typedef boost::shared_ptr < **DARTRayShape** > **DARTRayShapePtr**
- typedef boost::shared_ptr< **Entity** > **EntityPtr**
- typedef boost::shared_ptr < **Gripper** > **GripperPtr**
- typedef boost::shared_ptr < **HeightmapShape** > **HeightmapShapePtr**
- typedef boost::shared_ptr < **Inertial** > **InertialPtr**
- typedef std::vector< **JointPtr** > **Joint_V**
- typedef std::vector < **JointControllerPtr** > **JointController_V**
- typedef boost::shared_ptr < **JointController** > **JointControllerPtr**
- typedef boost::shared_ptr< **Joint** > **JointPtr**
- typedef std::map< std::string, **JointState** > **JointState_M**
- typedef std::vector< **LinkPtr** > **Link_V**
- typedef boost::shared_ptr< **Link** > **LinkPtr**
- typedef std::map< std::string, **LinkState** > **LinkState_M**
- typedef boost::shared_ptr < **MeshShape** > **MeshShapePtr**
- typedef std::vector< **ModelPtr** > **Model_V**
- typedef boost::shared_ptr< **Model** > **ModelPtr**
- typedef std::map< std::string, **ModelState** > **ModelState_M**

- typedef boost::shared_ptr < **MultiRayShape** > **MultiRayShapePtr**
- typedef boost::shared_ptr < **PhysicsEngine** > **PhysicsEnginePtr**
- typedef **PhysicsEnginePtr**(∗ **PhysicsFactoryFn** )(**WorldPtr** world)
- typedef boost::shared_ptr < **RayShape** > **RayShapePtr**
- typedef boost::shared_ptr< **Road** > **RoadPtr**
- typedef boost::shared_ptr< **Shape** > **ShapePtr**
- typedef boost::shared_ptr < **SimbodyCollision** > **SimbodyCollisionPtr**
- typedef boost::shared_ptr < **SimbodyLink** > **SimbodyLinkPtr**
- typedef boost::shared_ptr < **SimbodyModel** > **SimbodyModelPtr**
- typedef boost::shared_ptr < **SimbodyPhysics** > **SimbodyPhysicsPtr**
- typedef boost::shared_ptr < **SimbodyRayShape** > **SimbodyRayShapePtr**
- typedef boost::shared_ptr < **SphereShape** > **SphereShapePtr**
- typedef boost::shared_ptr < **SurfaceParams** > **SurfaceParamsPtr**
- typedef boost::shared_ptr< **World** > **WorldPtr**

## Functions

- **WorldPtr create_world** (const std::string &_name="")

  *Create a world given a name.*
- bool **fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 94).*
- **WorldPtr get_world** (const std::string &_name="")

  *Returns a pointer to a world by name.*
- uint32_t **getUniqueId** ()

  *Get a unique ID.*
- void **init_world** (**WorldPtr** _world)

  *Init world given a pointer to it.*
- void **init_worlds** ()

  *initialize multiple worlds stored in static variable gazebo::g_worlds*
- bool **load** ()

  *Setup **gazebo::SystemPlugin** (p. 1030)'s and call **gazebo::transport::init** (p. 95).*
- void **load_world** (**WorldPtr** _world, sdf::ElementPtr _sdf)

  *Load world from sdf::Element pointer.*
- void **load_worlds** (sdf::ElementPtr _sdf)

  *load multiple worlds from single sdf::Element pointer*
- void **pause_world** (**WorldPtr** _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 1167).*
- void **pause_worlds** (bool pause)

  *pause multiple worlds stored in static variable gazebo::g_worlds*

- void **remove_worlds** ()

  *remove multiple worlds stored in static variable gazebo::g_worlds*
- void **run_world** (**WorldPtr** _world, unsigned int _iterations=0)

  *Run world by calling **World::Run()** (p. 1167) given a pointer to it.*
- void **run_worlds** (unsigned int _iterations=0)

  *Run multiple worlds stored in static variable gazebo::g_worlds.*
- void **stop_world** (**WorldPtr** _world)

  *Stop world by calling **World::Stop()** (p. 1168) given a pointer to it.*
- void **stop_worlds** ()

  *stop multiple worlds stored in static variable gazebo::g_worlds*
- bool **worlds_running** ()

  *Return true if any world is running.*

### Variables

- static std::string **EntityTypename** []

  *String names for the different entity types.*

## 9.7.1 Detailed Description

namespace for physics Physics forward declarations and type defines.

physics namespace

## 9.7.2 Typedef Documentation

### 9.7.2.1 typedef std::vector<ActorPtr> gazebo::physics::Actor_V

### 9.7.2.2 typedef boost::shared_ptr<Actor> gazebo::physics::ActorPtr

### 9.7.2.3 typedef std::vector<BasePtr> gazebo::physics::Base_V

### 9.7.2.4 typedef boost::shared_ptr<Base> gazebo::physics::BasePtr

### 9.7.2.5 typedef boost::shared_ptr<BoxShape> gazebo::physics::BoxShapePtr

### 9.7.2.6 typedef std::vector<CollisionPtr> gazebo::physics::Collision_V

### 9.7.2.7 typedef boost::shared_ptr<Collision> gazebo::physics::CollisionPtr

### 9.7.2.8 typedef boost::shared_ptr<Contact> gazebo::physics::ContactPtr

### 9.7.2.9 typedef boost::shared_ptr<CylinderShape> gazebo::physics::CylinderShapePtr

### 9.7.2.10 typedef boost::shared_ptr<DARTCollision> gazebo::physics::DARTCollisionPtr

### 9.7.2.11 typedef boost::shared_ptr<DARTJoint> gazebo::physics::DARTJointPtr

### 9.7.2.12 typedef boost::shared_ptr<DARTLink> gazebo::physics::DARTLinkPtr

9.7.2.13 typedef boost::shared_ptr<DARTModel> gazebo::physics::DARTModelPtr

9.7.2.14 typedef boost::shared_ptr<DARTPhysics> gazebo::physics::DARTPhysicsPtr

9.7.2.15 typedef boost::shared_ptr<DARTRayShape> gazebo::physics::DARTRayShapePtr

9.7.2.16 typedef boost::shared_ptr<Entity> gazebo::physics::EntityPtr

9.7.2.17 typedef boost::shared_ptr<Gripper> gazebo::physics::GripperPtr

9.7.2.18 typedef boost::shared_ptr<HeightmapShape> gazebo::physics::HeightmapShapePtr

9.7.2.19 typedef boost::shared_ptr<Inertial> gazebo::physics::InertialPtr

9.7.2.20 typedef std::vector<JointPtr> gazebo::physics::Joint_V

9.7.2.21 typedef std::vector<JointControllerPtr> gazebo::physics::JointController_V

9.7.2.22 typedef boost::shared_ptr<JointController> gazebo::physics::JointControllerPtr

9.7.2.23 typedef boost::shared_ptr<Joint> gazebo::physics::JointPtr

9.7.2.24 typedef std::map<std::string, JointState> gazebo::physics::JointState_M

9.7.2.25 typedef std::vector<LinkPtr> gazebo::physics::Link_V

9.7.2.26 typedef boost::shared_ptr<Link> gazebo::physics::LinkPtr

9.7.2.27 typedef std::map<std::string, LinkState> gazebo::physics::LinkState_M

9.7.2.28 typedef boost::shared_ptr<MeshShape> gazebo::physics::MeshShapePtr

9.7.2.29 typedef std::vector<ModelPtr> gazebo::physics::Model_V

9.7.2.30 typedef boost::shared_ptr<Model> gazebo::physics::ModelPtr

9.7.2.31 typedef std::map<std::string, ModelState> gazebo::physics::ModelState_M

9.7.2.32 typedef boost::shared_ptr<MultiRayShape> gazebo::physics::MultiRayShapePtr

9.7.2.33 typedef boost::shared_ptr<PhysicsEngine> gazebo::physics::PhysicsEnginePtr

9.7.2.34 typedef boost::shared_ptr<RayShape> gazebo::physics::RayShapePtr

9.7.2.35 typedef boost::shared_ptr<Road> gazebo::physics::RoadPtr

9.7.2.36 typedef boost::shared_ptr<Shape> gazebo::physics::ShapePtr

9.7.2.37 typedef boost::shared_ptr<SimbodyCollision> gazebo::physics::SimbodyCollisionPtr

9.7.2.38 typedef boost::shared_ptr<SimbodyLink> gazebo::physics::SimbodyLinkPtr

#### 9.7.2.39 typedef boost::shared_ptr<SimbodyModel> gazebo::physics::SimbodyModelPtr

#### 9.7.2.40 typedef boost::shared_ptr<SimbodyPhysics> gazebo::physics::SimbodyPhysicsPtr

#### 9.7.2.41 typedef boost::shared_ptr<SimbodyRayShape> gazebo::physics::SimbodyRayShapePtr

#### 9.7.2.42 typedef boost::shared_ptr<SphereShape> gazebo::physics::SphereShapePtr

#### 9.7.2.43 typedef boost::shared_ptr<SurfaceParams> gazebo::physics::SurfaceParamsPtr

#### 9.7.2.44 typedef boost::shared_ptr<World> gazebo::physics::WorldPtr

## 9.8 gazebo::rendering Namespace Reference

Rendering namespace.

### Classes

- class **ArrowVisual**

  *Basic arrow visualization.*
- class **AxisVisual**

  *Basic axis visualization.*
- class **Camera**

  *Basic camera sensor.*
- class **CameraVisual**

  *Basic camera visualization.*
- class **COMVisual**

  *Basic Center of Mass visualization.*
- class **ContactVisual**

  *Contact visualization.*
- class **Conversions**

  ***Conversions*** *(*p. *273)* ***Conversions.hh*** *(*p. *1218)* ***rendering/Conversions.hh*** *(*p. *1218).*
- class **DepthCamera**

  *Depth camera used to render depth data into an image buffer.*
- class **DummyPageProvider**

  *Pretends to provide procedural page content to avoid page loading.*
- class **DynamicLines**

  *Class for drawing lines that can change.*
- class **DynamicRenderable**

  *Abstract base class providing mechanisms for dynamically growing hardware buffers.*
- class **Events**

  *Base class for rendering events.*
- class **FPSViewController**

  *First Person Shooter style view controller.*
- class **GpuLaser**

  *GPU based laser distance sensor.*
- class **Grid**

*Displays a grid of cells, drawn with lines.*

- class **GUIOverlay**

  *A class that creates a CEGUI overlay on a render window.*
- class **GzTerrainMatGen**
- class **Heightmap**

  *Rendering a terrain using heightmap information.*
- class **JointVisual**

  *Visualization for joints.*
- class **LaserVisual**

  *Visualization for laser data.*
- class **Light**

  *A light source.*
- class **MovableText**

  *Movable text.*
- class **OrbitViewController**

  *Orbit view controller.*
- class **Projector**

  *Projects a material onto surface, light a light projector.*
- class **RenderEngine**

  *Adaptor to Ogre3d.*
- class **RFIDTagVisual**

  *Visualization for RFID tags sensor.*
- class **RFIDVisual**

  *Visualization for RFID sensor.*
- class **Road2d**
- class **RTShaderSystem**

  *Implements **Ogre** (p. 130)'s Run-Time Shader system.*
- class **Scene**

  *Representation of an entire scene graph.*
- class **SelectionObj**

  *Interactive selection object for models and links.*
- class **SonarVisual**

  *Visualization for sonar data.*
- class **TransmitterVisual**

  *Visualization for the wireless propagation data.*
- class **UserCamera**

  *A camera used for user visualization of a scene.*
- class **VideoVisual**

  *A visual element that displays a video as a texture.*
- class **ViewController**

  *Base class for view controllers.*
- class **Visual**

  *A renderable object.*
- class **WindowManager**

  *Class to mangage render windows.*
- class **WireBox**

  *Draws a wireframe box.*
- class **WrenchVisual**

  *Visualization for sonar data.*

## Typedefs

- typedef boost::shared_ptr < **ArrowVisual** > **ArrowVisualPtr**
- typedef boost::shared_ptr < **AxisVisual** > **AxisVisualPtr**
- typedef boost::shared_ptr< **Camera** > **CameraPtr**
- typedef boost::shared_ptr < **CameraVisual** > **CameraVisualPtr**
- typedef boost::shared_ptr < **COMVisual** > **COMVisualPtr**
- typedef boost::shared_ptr < **ContactVisual** > **ContactVisualPtr**
- typedef boost::shared_ptr < **DepthCamera** > **DepthCameraPtr**
- typedef boost::shared_ptr < **DynamicLines** > **DynamicLinesPtr**
- typedef boost::shared_ptr < **GpuLaser** > **GpuLaserPtr**
- typedef boost::shared_ptr < **JointVisual** > **JointVisualPtr**
- typedef boost::shared_ptr < **LaserVisual** > **LaserVisualPtr**
- typedef boost::shared_ptr< **Light** > **LightPtr**
- typedef boost::shared_ptr < **RFIDTagVisual** > **RFIDTagVisualPtr**
- typedef boost::shared_ptr < **RFIDVisual** > **RFIDVisualPtr**
- typedef boost::shared_ptr< **Scene** > **ScenePtr**
- typedef boost::shared_ptr < **SelectionObj** > **SelectionObjPtr**
- typedef boost::shared_ptr < **SonarVisual** > **SonarVisualPtr**
- typedef boost::shared_ptr < **UserCamera** > **UserCameraPtr**
- typedef boost::shared_ptr< **Visual** > **VisualPtr**
- typedef boost::shared_ptr < **WindowManager** > **WindowManagerPtr**
- typedef boost::shared_ptr < **WrenchVisual** > **WrenchVisualPtr**

## Enumerations

- enum **RenderOpType** { **RENDERING_POINT_LIST** = 0, **RENDERING_LINE_LIST** = 1, **RENDERING_LINE_STRIP** = 2, **RENDERING_TRIANGLE_LIST** = 3, **RENDERING_TRIANGLE_STRIP** = 4, **RENDERING_TRIANGLE_FAN** = 5, **RENDERING_MESH_RESOURCE** = 6 }

  *Type of render operation for a drawable.*

### Functions

- **rendering::ScenePtr create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)

  *create **rendering::Scene*** (p. *814*) *by name.*
- bool **fini** ()

  *teardown rendering engine.*
- **rendering::ScenePtr get_scene** (const std::string &_name="")

  *get pointer to **rendering::Scene*** (p. *814*) *by name.*
- bool **init** ()

  *init rendering engine.*
- bool **load** ()

  *load rendering engine.*
- void **remove_scene** (const std::string &_name)

  *remove a **rendering::Scene*** (p. *814*) *by name*

## 9.8.1 Detailed Description

Rendering namespace.

## 9.8.2 Typedef Documentation

### 9.8.2.1 typedef boost::shared_ptr<ArrowVisual> gazebo::rendering::ArrowVisualPtr

### 9.8.2.2 typedef boost::shared_ptr<AxisVisual> gazebo::rendering::AxisVisualPtr

### 9.8.2.3 typedef boost::shared_ptr<Camera> gazebo::rendering::CameraPtr

### 9.8.2.4 typedef boost::shared_ptr<CameraVisual> gazebo::rendering::CameraVisualPtr

### 9.8.2.5 typedef boost::shared_ptr<COMVisual> gazebo::rendering::COMVisualPtr

### 9.8.2.6 typedef boost::shared_ptr<ContactVisual> gazebo::rendering::ContactVisualPtr

### 9.8.2.7 typedef boost::shared_ptr<DepthCamera> gazebo::rendering::DepthCameraPtr

### 9.8.2.8 typedef boost::shared_ptr<DynamicLines> gazebo::rendering::DynamicLinesPtr

### 9.8.2.9 typedef boost::shared_ptr<GpuLaser> gazebo::rendering::GpuLaserPtr

### 9.8.2.10 typedef boost::shared_ptr<JointVisual> gazebo::rendering::JointVisualPtr

### 9.8.2.11 typedef boost::shared_ptr<LaserVisual> gazebo::rendering::LaserVisualPtr

### 9.8.2.12 typedef boost::shared_ptr<Light> gazebo::rendering::LightPtr

### 9.8.2.13 typedef boost::shared_ptr<RFIDTagVisual> gazebo::rendering::RFIDTagVisualPtr

### 9.8.2.14 typedef boost::shared_ptr<RFIDVisual> gazebo::rendering::RFIDVisualPtr

#### 9.8.2.15 typedef boost::shared_ptr<Scene> gazebo::rendering::ScenePtr

#### 9.8.2.16 typedef boost::shared_ptr<SelectionObj> gazebo::rendering::SelectionObjPtr

#### 9.8.2.17 typedef boost::shared_ptr<SonarVisual> gazebo::rendering::SonarVisualPtr

#### 9.8.2.18 typedef boost::shared_ptr<UserCamera> gazebo::rendering::UserCameraPtr

#### 9.8.2.19 typedef boost::shared_ptr<Visual> gazebo::rendering::VisualPtr

#### 9.8.2.20 typedef boost::shared_ptr<WindowManager> gazebo::rendering::WindowManagerPtr

#### 9.8.2.21 typedef boost::shared_ptr<WrenchVisual> gazebo::rendering::WrenchVisualPtr

### 9.8.3 Enumeration Type Documentation

#### 9.8.3.1 enum gazebo::rendering::RenderOpType

Type of render operation for a drawable.

**Enumerator**

***RENDERING_POINT_LIST*** A list of points, 1 vertex per point.

***RENDERING_LINE_LIST*** A list of lines, 2 vertices per line.

***RENDERING_LINE_STRIP*** A strip of connected lines, 1 vertex per line plus 1 start vertex.

***RENDERING_TRIANGLE_LIST*** A list of triangles, 3 vertices per triangle.

***RENDERING_TRIANGLE_STRIP*** A strip of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

***RENDERING_TRIANGLE_FAN*** A fan of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

***RENDERING_MESH_RESOURCE*** N/A.

## 9.9 gazebo::sensors Namespace Reference

Sensors namespace.

### Classes

- class **CameraSensor**

  *Basic camera sensor.*
- class **ContactSensor**

  *Contact sensor.*
- class **DepthCameraSensor**
- class **ForceTorqueSensor**

  ***Sensor*** *(p. 837) for measure force and torque on a joint.*
- class **GpsSensor**

  ***GpsSensor*** *(p. 426) to provide position measurement.*
- class **GpuRaySensor**
- class **ImuSensor**

  *An IMU sensor.*

- class **MultiCameraSensor**

  *Multiple camera sensor.*
- class **Noise**

  ***Noise*** *(p. 689) models for sensor output signals.*
- class **RaySensor**

  ***Sensor*** *(p. 837) with one or more rays.*
- class **RFIDSensor**

  ***Sensor*** *(p. 837) class for RFID type of sensor.*
- class **RFIDTag**

  ***RFIDTag*** *(p. 798) to interact with RFIDTagSensors.*
- class **Sensor**

  *Base class for sensors.*
- class **SensorFactory**
- class **SensorManager**

  *Class to manage and update all sensors.*
- class **SonarSensor**

  ***Sensor*** *(p. 837) with sonar cone.*
- class **WirelessReceiver**

  ***Sensor*** *(p. 837) class for receiving wireless signals.*
- class **WirelessTransceiver**

  ***Sensor*** *(p. 837) class for receiving wireless signals.*
- class **WirelessTransmitter**

  *Transmitter to send wireless signals.*

## Typedefs

- typedef std::vector
  < **CameraSensorPtr** > **CameraSensor_V**
- typedef boost::shared_ptr
  < **CameraSensor** > **CameraSensorPtr**
- typedef std::vector
  < **ContactSensorPtr** > **ContactSensor_V**
- typedef boost::shared_ptr
  < **ContactSensor** > **ContactSensorPtr**
- typedef std::vector
  < **DepthCameraSensorPtr** > **DepthCameraSensor_V**
- typedef boost::shared_ptr
  < **DepthCameraSensor** > **DepthCameraSensorPtr**
- typedef boost::shared_ptr
  < **ForceTorqueSensor** > **ForceTorqueSensorPtr**
- typedef boost::shared_ptr
  < **GpsSensor** > **GpsSensorPtr**
- typedef std::vector
  < **GpuRaySensorPtr** > **GpuRaySensor_V**
- typedef boost::shared_ptr
  < **GpuRaySensor** > **GpuRaySensorPtr**
- typedef std::vector< **ImuSensorPtr** > **ImuSensor_V**
- typedef boost::shared_ptr
  < **ImuSensor** > **ImuSensorPtr**

- typedef std::vector < **MultiCameraSensorPtr** > **MultiCameraSensor_V**
- typedef boost::shared_ptr < **MultiCameraSensor** > **MultiCameraSensorPtr**
- typedef boost::shared_ptr< **Noise** > **NoisePtr**
- typedef std::vector< **RaySensorPtr** > **RaySensor_V**
- typedef boost::shared_ptr < **RaySensor** > **RaySensorPtr**
- typedef std::vector< **RFIDSensor** > **RFIDSensor_V**
- typedef boost::shared_ptr < **RFIDSensor** > **RFIDSensorPtr**
- typedef std::vector< **RFIDTag** > **RFIDTag_V**
- typedef boost::shared_ptr < **RFIDTag** > **RFIDTagPtr**
- typedef std::vector< **SensorPtr** > **Sensor_V**
- typedef **Sensor** *(* **SensorFactoryFn** )()
- typedef boost::shared_ptr< **Sensor** > **SensorPtr**
- typedef boost::shared_ptr < **SonarSensor** > **SonarSensorPtr**
- typedef std::vector < **WirelessReceiver** > **WirelessReceiver_V**
- typedef boost::shared_ptr < **WirelessReceiver** > **WirelessReceiverPtr**
- typedef std::vector < **WirelessTransceiver** > **WirelessTransceiver_V**
- typedef boost::shared_ptr < **WirelessTransceiver** > **WirelessTransceiverPtr**
- typedef std::vector < **WirelessTransmitter** > **WirelessTransmitter_V**
- typedef boost::shared_ptr < **WirelessTransmitter** > **WirelessTransmitterPtr**

## Enumerations

- enum **SensorCategory** { **IMAGE** = 0, **RAY** = 1, **OTHER** = 2, **CATEGORY_COUNT** = 3 }

  *SensorClass is used to categorize sensors.*

## Functions

- std::string **create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(2.0)

  *Deprecated.*
- std::string **create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  *Create a sensor using SDF.*
- void **disable** ()

  *Disable sensors.*
- void **enable** ()

  *Enable sensors.*
- bool **fini** ()

*shutdown the sensor generation loop.*

- **SensorPtr get_sensor** (const std::string &_name)

  *Get a sensor using by name.*

- bool **init** ()

  *initialize the sensor generation loop.*

- bool **load** ()

  *Load the sensor library.*

- void **remove_sensor** (const std::string &_sensorName)

  *Remove a sensor by name.*

- bool **remove_sensors** ()

  *Remove all sensors.*

- void **run_once** (bool _force=false)

  *Run the sensor generation one step.*

- void **run_threads** ()

  *Run sensors in a threads. This is a non-blocking call.*

- void **stop** ()

  *Stop the sensor generation loop.*

### 9.9.1 Detailed Description

Sensors namespace.

### 9.9.2 Typedef Documentation

#### 9.9.2.1 typedef std::vector<CameraSensorPtr> gazebo::sensors::CameraSensor_V

#### 9.9.2.2 typedef boost::shared_ptr<CameraSensor> gazebo::sensors::CameraSensorPtr

#### 9.9.2.3 typedef std::vector<ContactSensorPtr> gazebo::sensors::ContactSensor_V

#### 9.9.2.4 typedef boost::shared_ptr<ContactSensor> gazebo::sensors::ContactSensorPtr

#### 9.9.2.5 typedef std::vector<DepthCameraSensorPtr> gazebo::sensors::DepthCameraSensor_V

#### 9.9.2.6 typedef boost::shared_ptr<DepthCameraSensor> gazebo::sensors::DepthCameraSensorPtr

#### 9.9.2.7 typedef boost::shared_ptr<ForceTorqueSensor> gazebo::sensors::ForceTorqueSensorPtr

#### 9.9.2.8 typedef boost::shared_ptr<GpsSensor> gazebo::sensors::GpsSensorPtr

#### 9.9.2.9 typedef std::vector<GpuRaySensorPtr> gazebo::sensors::GpuRaySensor_V

#### 9.9.2.10 typedef boost::shared_ptr<GpuRaySensor> gazebo::sensors::GpuRaySensorPtr

#### 9.9.2.11 typedef std::vector<ImuSensorPtr> gazebo::sensors::ImuSensor_V

#### 9.9.2.12 typedef boost::shared_ptr<ImuSensor> gazebo::sensors::ImuSensorPtr

#### 9.9.2.13 typedef std::vector<MultiCameraSensorPtr> gazebo::sensors::MultiCameraSensor_V

#### 9.9.2.14 typedef boost::shared_ptr<MultiCameraSensor> gazebo::sensors::MultiCameraSensorPtr

#### 9.9.2.15 typedef boost::shared_ptr<Noise> gazebo::sensors::NoisePtr

#### 9.9.2.16 typedef std::vector<RaySensorPtr> gazebo::sensors::RaySensor_V

#### 9.9.2.17 typedef boost::shared_ptr<RaySensor> gazebo::sensors::RaySensorPtr

#### 9.9.2.18 typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V

#### 9.9.2.19 typedef boost::shared_ptr<RFIDSensor> gazebo::sensors::RFIDSensorPtr

#### 9.9.2.20 typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V

#### 9.9.2.21 typedef boost::shared_ptr<RFIDTag> gazebo::sensors::RFIDTagPtr

#### 9.9.2.22 typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V

#### 9.9.2.23 typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()

#### 9.9.2.24 typedef boost::shared_ptr<Sensor> gazebo::sensors::SensorPtr

#### 9.9.2.25 typedef boost::shared_ptr<SonarSensor> gazebo::sensors::SonarSensorPtr

#### 9.9.2.26 typedef std::vector<WirelessReceiver> gazebo::sensors::WirelessReceiver_V

#### 9.9.2.27 typedef boost::shared_ptr<WirelessReceiver> gazebo::sensors::WirelessReceiverPtr

#### 9.9.2.28 typedef std::vector<WirelessTransceiver> gazebo::sensors::WirelessTransceiver_V

#### 9.9.2.29 typedef boost::shared_ptr<WirelessTransceiver> gazebo::sensors::WirelessTransceiverPtr

#### 9.9.2.30 typedef std::vector<WirelessTransmitter> gazebo::sensors::WirelessTransmitter_V

#### 9.9.2.31 typedef boost::shared_ptr<WirelessTransmitter> gazebo::sensors::WirelessTransmitterPtr

### 9.9.3 Enumeration Type Documentation

#### 9.9.3.1 enum gazebo::sensors::SensorCategory

SensorClass is used to categorize sensors.

This is used to put sensors into different threads.

**Enumerator**

***IMAGE*** Image based sensor class. This type requires the rendering engine.

***RAY*** Ray based sensor class.

***OTHER*** A type of sensor is not a RAY or IMAGE sensor.

***CATEGORY_COUNT*** Number of **Sensor** (p. 837) Categories.

## 9.10 gazebo::transport Namespace Reference

### Classes

- class **CallbackHelper**

  *A helper class to handle callbacks when messages arrive.*
- class **CallbackHelperT**

  *Callback helper Template.*
- class **Connection**

  *Single TCP/IP connection manager.*
- class **ConnectionManager**

  *Manager of connections.*
- class **IOManager**

  *Manages boost::asio IO.*
- class **Node**

  *A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*
- class **Publication**

  *A publication for a topic.*
- class **PublicationTransport**

  *transport/transport.hh*
- class **Publisher**

  *A publisher of messages on a topic.*
- class **RawCallbackHelper**

  *Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.*
- class **SubscribeOptions**

  *Options for a subscription.*
- class **Subscriber**

  *A subscriber to a topic.*
- class **SubscriptionTransport**

  *transport/transport.hh*
- class **TopicManager**

  *Manages topics and their subscriptions.*

### Typedefs

- typedef boost::shared_ptr
  < **CallbackHelper** > **CallbackHelperPtr**

  *boost shared pointer to* ***transport::CallbackHelper*** *(*p. *180)*
- typedef boost::shared_ptr
  < **Connection** > **ConnectionPtr**
- typedef boost::shared_ptr
  < google::protobuf::Message > **MessagePtr**
- typedef boost::shared_ptr< **Node** > **NodePtr**
- typedef boost::shared_ptr
  < **Publication** > **PublicationPtr**
- typedef boost::shared_ptr
  < **PublicationTransport** > **PublicationTransportPtr**

- typedef boost::shared_ptr < **Publisher** > **PublisherPtr**
- typedef boost::shared_ptr < **Subscriber** > **SubscriberPtr**
- typedef boost::shared_ptr < **SubscriptionTransport** > **SubscriptionTransportPtr**

## Functions

- void **clear_buffers** ()

  *Clear any remaining communication buffers.*
- void **fini** ()

  *Cleanup the transport component.*
- bool **get_master_uri** (std::string &_master_host, unsigned int &_master_port)

  *Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.*
- void **get_topic_namespaces** (std::list< std::string > &_namespaces)

  *Return all the namespace (world names) on the master.*
- std::map< std::string, std::list< std::string > > **getAdvertisedTopics** ()

  *Get a list of all the topics and their message types.*
- std::list< std::string > **getAdvertisedTopics** (const std::string &_msgType)

  *Get a list of all the unique advertised topic names.*
- bool **getMinimalComms** ()

  *Get whether minimal comms has been enabled.*
- std::string **getTopicMsgType** (const std::string &_topicName)

  *Get the message typename that is published on the given topic.*
- bool **init** (const std::string &_master_host="", unsigned int _master_port=0)

  *Initialize the transport system.*
- bool **is_stopped** ()

  *Is the transport system stopped?*
- void **pause_incoming** (bool _pause)

  *Pause or unpause incoming messages.*
- template<typename M > void **publish** (const std::string &_topic, const google::protobuf::Message &_message)

  *A convenience function for a one-time publication of a message.*
- boost::shared_ptr< msgs::Response > **request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  *Send a request and receive a response.*
- void **requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  *Send a request and don't wait for a response.*
- void **requestNoReply** (**NodePtr** _node, const std::string &_request, const std::string &_data="")

  *Send a request and don't wait for a response.*
- void **run** ()

  *Run the transport component.*
- void **setMinimalComms** (bool _enabled)

  *Set whether minimal comms should be used.*
- void **stop** ()

  *Stop the transport component from running.*

### 9.10.1 Typedef Documentation

#### 9.10.1.1 typedef boost::shared_ptr<Connection> gazebo::transport::ConnectionPtr

#### 9.10.1.2 typedef boost::shared_ptr<google::protobuf::Message> gazebo::transport::MessagePtr

#### 9.10.1.3 typedef boost::shared_ptr<Node> gazebo::transport::NodePtr

#### 9.10.1.4 typedef boost::shared_ptr<Publication> gazebo::transport::PublicationPtr

#### 9.10.1.5 typedef boost::shared_ptr<PublicationTransport> gazebo::transport::PublicationTransportPtr

#### 9.10.1.6 typedef boost::shared_ptr<Publisher> gazebo::transport::PublisherPtr

#### 9.10.1.7 typedef boost::shared_ptr<Subscriber> gazebo::transport::SubscriberPtr

#### 9.10.1.8 typedef boost::shared_ptr<SubscriptionTransport> gazebo::transport::SubscriptionTransportPtr

## 9.11 gazebo::util Namespace Reference

### Classes

- class **DiagnosticManager**

  *A diagnostic manager class.*
- class **DiagnosticTimer**

  *A timer designed for diagnostics.*
- class **LogPlay**
- class **LogRecord**

  *addtogroup gazebo_util*
- class **OpenAL**

  *3D audio setup and playback.*
- class **OpenALSink**

  ***OpenAL*** *(*p. *695) Listener.*
- class **OpenALSource**

  ***OpenAL*** *(*p. *695) Source.*

### Typedefs

- typedef boost::shared_ptr < **DiagnosticTimer** > **DiagnosticTimerPtr**
- typedef boost::shared_ptr < **OpenALSink** > **OpenALSinkPtr**
- typedef boost::shared_ptr < **OpenALSource** > **OpenALSourcePtr**

### 9.11.1 Typedef Documentation

#### 9.11.1.1 typedef boost::shared_ptr<DiagnosticTimer> gazebo::util::DiagnosticTimerPtr

#### 9.11.1.2 typedef boost::shared_ptr<OpenALSink> gazebo::util::OpenALSinkPtr

#### 9.11.1.3 typedef boost::shared_ptr<OpenALSource> gazebo::util::OpenALSourcePtr

## 9.12 google Namespace Reference

### Namespaces

- namespace **protobuf**

## 9.13 google::protobuf Namespace Reference

### Namespaces

- namespace **compiler**

## 9.14 google::protobuf::compiler Namespace Reference

### Namespaces

- namespace **cpp**

## 9.15 google::protobuf::compiler::cpp Namespace Reference

### Classes

- class **GazeboGenerator**

  *Google protobuf message generator for* ***gazebo::msgs*** *(*p. *108**).*

## 9.16 Ogre Namespace Reference

## 9.17 ogre Namespace Reference

## 9.18 SimTK Namespace Reference

## 9.19 SkyX Namespace Reference

# Chapter 10

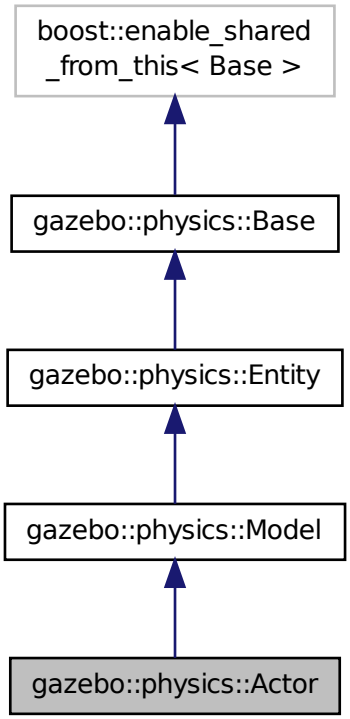
# Class Documentation

## 10.1 gazebo::physics::Actor Class Reference

**Actor** (p. 131) class enables GPU based mesh model / skeleton scriptable animation.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Actor:



## Public Member Functions

- **Actor** (**BasePtr** _parent)

  *Constructor.*
- virtual ∼**Actor** ()

  *Destructor.*
- virtual void **Fini** ()

  *Finalize the actor.*
- virtual const sdf::ElementPtr **GetSDF** ()

  *Get the SDF values for the actor.*
- virtual void **Init** ()

  *Initialize the actor.*
- virtual bool **IsActive** ()

  *Returns true when actor is playing animation.*
- void **Load** (sdf::ElementPtr _sdf)

  *Load the actor.*
- virtual void **Play** ()

  *Start playing the script.*
- virtual void **Stop** ()

  *Stop playing the script.*
- void **Update** ()

  *Update the actor.*
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  *update the parameters using new sdf values.*

## Protected Attributes

- bool **active**

  *True if the actor is being updated.*
- bool **autoStart**

  *True if the actor should start running automatically.*
- **transport::PublisherPtr bonePosePub**

  *Where to send bone info.*
- std::map< std::string, bool > **interpolateX**

  *True to interpolate along x direction.*
- **math::Vector3 lastPos**

  *Last position of the actor.*
- double **lastScriptTime**

  *Time the scipt was last updated.*
- unsigned int **lastTraj**

  *THe last trajectory.*
- bool **loop**

  *True if the animation should loop.*
- **LinkPtr mainLink**

  ***Base*** *(p. 159) link.*
- const **common::Mesh** ∗ **mesh**

  *Pointer to the actor's mesh.*

- std::string **oldAction**

  *THe old action.*
- double **pathLength**

  *Length of the actor's path.*
- **common::Time playStartTime**

  *Time when the animation was started.*
- **common::Time prevFrameTime**

  *Time of the previous frame.*
- double **scriptLength**

  *Time length of a scipt.*
- std::map< std::string, **common::SkeletonAnimation** * > **skelAnimation**

  *Skeleton animations.*
- **common::Skeleton** * **skeleton**

  *The actor's skeleton.*
- std::map< std::string, std::map< std::string, std::string > > **skelNodesMap**

  *Skeleton to naode map.*
- std::string **skinFile**

  *Filename for the skin.*
- double **skinScale**

  *Scaling factor to apply to the skin.*
- double **startDelay**

  *Amount of time to delay start by.*
- std::map< unsigned int, **common::PoseAnimation** * > **trajectories**

  *All the trajectories.*
- std::vector< **TrajectoryInfo** > **trajInfo**

  *Trajectory information.*
- uint32_t **visualId**

  *ID for this visual.*
- std::string **visualName**

  *Name of the visual.*

## Additional Inherited Members

### 10.1.1 Detailed Description

**Actor** (p. 131) class enables GPU based mesh model / skeleton scriptable animation.

### 10.1.2 Constructor & Destructor Documentation

#### 10.1.2.1 gazebo::physics::Actor::Actor ( **BasePtr** *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent object |

#### 10.1.2.2 virtual gazebo::physics::Actor::~Actor ( ) [virtual]

Destructor.

### 10.1.3 Member Function Documentation

#### 10.1.3.1 virtual void gazebo::physics::Actor::Fini ( ) [virtual]

Finalize the actor.

Reimplemented from **gazebo::physics::Model** (p. 629).

#### 10.1.3.2 virtual const sdf::ElementPtr gazebo::physics::Actor::GetSDF ( ) [virtual]

Get the SDF values for the actor.

**Returns**

Pointer to the SDF values.

Reimplemented from **gazebo::physics::Model** (p. 632).

#### 10.1.3.3 virtual void gazebo::physics::Actor::Init ( ) [virtual]

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 633).

#### 10.1.3.4 virtual bool gazebo::physics::Actor::IsActive ( ) [virtual]

Returns true when actor is playing animation.

#### 10.1.3.5 void gazebo::physics::Actor::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the actor.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters |

Reimplemented from **gazebo::physics::Entity** (p. 386).

#### 10.1.3.6 virtual void gazebo::physics::Actor::Play ( ) [virtual]

Start playing the script.

#### 10.1.3.7 virtual void gazebo::physics::Actor::Stop ( ) [virtual]

Stop playing the script.

#### 10.1.3.8 void gazebo::physics::Actor::Update ( ) [virtual]

Update the actor.

Reimplemented from **gazebo::physics::Base** (p. 171).

#### 10.1.3.9 virtual void gazebo::physics::Actor::UpdateParameters ( sdf::ElementPtr _sdf ) [virtual]

update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to update from. |

Reimplemented from **gazebo::physics::Model** (p. 638).

### 10.1.4 Member Data Documentation

#### 10.1.4.1 bool gazebo::physics::Actor::active [protected]

True if the actor is being updated.

#### 10.1.4.2 bool gazebo::physics::Actor::autoStart [protected]

True if the actor should start running automatically.

#### 10.1.4.3 transport::PublisherPtr gazebo::physics::Actor::bonePosePub [protected]

Where to send bone info.

#### 10.1.4.4 std::map<std::string, bool> gazebo::physics::Actor::interpolateX [protected]

True to interpolate along x direction.

#### 10.1.4.5 math::Vector3 gazebo::physics::Actor::lastPos [protected]

Last position of the actor.

#### 10.1.4.6 double gazebo::physics::Actor::lastScriptTime [protected]

Time the scipt was last updated.

#### 10.1.4.7 unsigned int gazebo::physics::Actor::lastTraj [protected]

THe last trajectory.

#### 10.1.4.8 bool gazebo::physics::Actor::loop [protected]

True if the animation should loop.

#### 10.1.4.9 LinkPtr gazebo::physics::Actor::mainLink [protected]

**Base** (p. 159) link.

#### 10.1.4.10 const common::Mesh* gazebo::physics::Actor::mesh [protected]

Pointer to the actor's mesh.

#### 10.1.4.11 std::string gazebo::physics::Actor::oldAction [protected]

THe old action.

#### 10.1.4.12 double gazebo::physics::Actor::pathLength [protected]

Length of the actor's path.

#### 10.1.4.13 common::Time gazebo::physics::Actor::playStartTime [protected]

Time when the animation was started.

#### 10.1.4.14 common::Time gazebo::physics::Actor::prevFrameTime [protected]

Time of the previous frame.

#### 10.1.4.15 double gazebo::physics::Actor::scriptLength [protected]

Time length of a scipt.

#### 10.1.4.16 std::map<std::string, common::SkeletonAnimation*> gazebo::physics::Actor::skelAnimation [protected]

Skeleton animations.

#### 10.1.4.17 common::Skeleton* gazebo::physics::Actor::skeleton [protected]

The actor's skeleton.

#### 10.1.4.18 std::map<std::string, std::map<std::string, std::string> > gazebo::physics::Actor::skelNodesMap [protected]

Skeleton to naode map.

#### 10.1.4.19 std::string gazebo::physics::Actor::skinFile [protected]

Filename for the skin.

#### 10.1.4.20 double gazebo::physics::Actor::skinScale [protected]

Scaling factor to apply to the skin.

#### 10.1.4.21 double gazebo::physics::Actor::startDelay [protected]

Amount of time to delay start by.

#### 10.1.4.22 std::map<unsigned int, common::PoseAnimation*> gazebo::physics::Actor::trajectories [protected]

All the trajectories.

#### 10.1.4.23 std::vector<TrajectoryInfo> gazebo::physics::Actor::trajInfo [protected]

Trajectory information.

#### 10.1.4.24 uint32_t gazebo::physics::Actor::visualId [protected]

ID for this visual.

#### 10.1.4.25 std::string gazebo::physics::Actor::visualName [protected]

Name of the visual.

The documentation for this class was generated from the following file:

- **Actor.hh**

## 10.2 gazebo::math::Angle Class Reference

An angle and related functions.

`#include <math/gzmath.hh>`

### Public Member Functions

- **Angle** ()
  *Constructor.*
- **Angle** (double _radian)

*Copy Constructor.*

- **Angle** (const **Angle** &_angle)

  *Copy constructor.*

- virtual ~**Angle** ()

  *Destructor.*

- double **Degree** () const

  *Get the angle in degrees.*

- void **Normalize** ()

  *Normalize the angle in the range -Pi to Pi.*

- bool **operator!=** (const **Angle** &_angle) const

  *Inequality.*

- double **operator*** () const

  *Dereference operator.*

- **Angle operator*** (const **Angle** &_angle) const

  *Multiplication operator, result = this * _angle.*

- **Angle operator*=** (const **Angle** &_angle)

  *Multiplication set, this = this * _angle.*

- **Angle operator+** (const **Angle** &_angle) const

  *Addition operator, result = this + _angle.*

- **Angle operator+=** (const **Angle** &_angle)

  *Addition set, this = this + _angle.*

- **Angle operator-** (const **Angle** &_angle) const

  *Substraction, result = this - _angle.*

- **Angle operator-=** (const **Angle** &_angle)

  *Subtraction set, this = this - _angle.*

- **Angle operator/** (const **Angle** &_angle) const

  *Division, result = this / _angle.*

- **Angle operator/=** (const **Angle** &_angle)

  *Division set, this = this / _angle.*

- bool **operator<** (const **Angle** &_angle) const

  *Less than operator.*

- bool **operator<=** (const **Angle** &_angle) const

  *Less or equal operator.*

- bool **operator==** (const **Angle** &_angle) const

  *Equality operator, result = this == _angle.*

- bool **operator>** (const **Angle** &_angle) const

  *Greater than operator.*

- bool **operator>=** (const **Angle** &_angle) const

  *Greater or equal operator.*

- double **Radian** () const

  *Get the angle in radians.*

- void **SetFromDegree** (double _degree)

  *Set the value from an angle in degrees.*

- void **SetFromRadian** (double _radian)

  *Set the value from an angle in radians.*

### Static Public Attributes

- static const **Angle HalfPi**

  ***math::Angle*** *(p. 137)(M_PI ∗ 0.5)*
- static const **Angle Pi**

  *math::Angle(M_PI)*
- static const **Angle TwoPi**

  *math::Angle(M_PI ∗ 2)*
- static const **Angle Zero**

  *math::Angle(0)*

### Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Angle** &_a)

  *Stream insertion operator.*
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Angle** &_a)

  *Stream extraction operator.*

## 10.2.1 Detailed Description

An angle and related functions.

## 10.2.2 Constructor & Destructor Documentation

#### 10.2.2.1 gazebo::math::Angle::Angle ( )

Constructor.

#### 10.2.2.2 gazebo::math::Angle::Angle ( double *_radian* )

Copy Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_radian* | Radians |

#### 10.2.2.3 gazebo::math::Angle::Angle ( const Angle & *_angle* )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) to copy |

#### 10.2.2.4 virtual gazebo::math::Angle::∼Angle ( ) [virtual]

Destructor.

### 10.2.3 Member Function Documentation

#### 10.2.3.1 double gazebo::math::Angle::Degree ( ) const

Get the angle in degrees.

**Returns**

double containing the angle's degree value

#### 10.2.3.2 void gazebo::math::Angle::Normalize ( )

Normalize the angle in the range -Pi to Pi.

#### 10.2.3.3 bool gazebo::math::Angle::operator!= ( const **Angle** & *_angle* ) const

Inequality.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) to check for inequality |

**Returns**

true if this != _angle

#### 10.2.3.4 double gazebo::math::Angle::operator* ( ) const [inline]

Dereference operator.

**Returns**

Double containing the angle's radian value

#### 10.2.3.5 **Angle** gazebo::math::Angle::operator* ( const **Angle** & *_angle* ) const

Multiplication operator, result = this * _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for multiplication |

**Returns**

the new angle

#### 10.2.3.6 **Angle** gazebo::math::Angle::operator*= ( const **Angle** & *_angle* )

Multiplication set, this = this * _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for multiplication |

**Returns**

angle

#### 10.2.3.7 Angle gazebo::math::Angle::operator+ ( const Angle & *_angle* ) const

Addition operator, result = this + _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for addition |

**Returns**

the new angle

#### 10.2.3.8 Angle gazebo::math::Angle::operator+= ( const Angle & *_angle* )

Addition set, this = this + _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for addition |

**Returns**

angle

#### 10.2.3.9 Angle gazebo::math::Angle::operator- ( const Angle & *_angle* ) const

Substraction, result = this - _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for substraction |

**Returns**

the new angle

#### 10.2.3.10 Angle gazebo::math::Angle::operator-= ( const Angle & *_angle* )

Subtraction set, this = this - _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for subtraction |

**Returns**

angle

#### 10.2.3.11 Angle gazebo::math::Angle::operator/ ( const Angle & *_angle* ) const

Division, result = this / _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for division |

**Returns**

the new angle

#### 10.2.3.12 Angle gazebo::math::Angle::operator/= ( const Angle & *_angle* )

Division set, this = this / _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) for division |

**Returns**

angle

#### 10.2.3.13 bool gazebo::math::Angle::operator< ( const Angle & *_angle* ) const

Less than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) to check |

**Returns**

true if this < _angle

#### 10.2.3.14 bool gazebo::math::Angle::operator<= ( const Angle & *_angle* ) const

Less or equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) to check |

**Returns**

true if this <= _angle

#### 10.2.3.15 bool gazebo::math::Angle::operator== ( const Angle & *_angle* ) const

Equality operator, result = this == _angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) to check for equality |

**Returns**

true if this == _angle

#### 10.2.3.16 bool gazebo::math::Angle::operator> ( const Angle & *_angle* ) const

Greater than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) to check |

**Returns**

true if this > _angle

#### 10.2.3.17 bool gazebo::math::Angle::operator>= ( const Angle & *_angle* ) const

Greater or equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | **Angle** (p. 137) to check |

**Returns**

true if this >= _angle

#### 10.2.3.18 double gazebo::math::Angle::Radian ( ) const

Get the angle in radians.

**Returns**

double containing the angle's radian value

#### 10.2.3.19 void gazebo::math::Angle::SetFromDegree ( double *_degree* )

Set the value from an angle in degrees.

**Parameters**

| | | |
|---|---|---|
| in | *_degree* | Degree value |

#### 10.2.3.20 void gazebo::math::Angle::SetFromRadian ( double *_radian* )

Set the value from an angle in radians.

**Parameters**

| | | |
|---|---|---|
| in | *_radian* | Radian value |

### 10.2.4 Friends And Related Function Documentation

#### 10.2.4.1 std::ostream& operator<< ( std::ostream & *_out,* const gazebo::math::Angle & *_a* ) [friend]

Stream insertion operator.

Outputs in degrees

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_a* | angle to output |

**Returns**

The output stream

#### 10.2.4.2 std::istream& operator>> ( std::istream & *_in,* gazebo::math::Angle & *_a* ) [friend]

Stream extraction operator.

Assumes input is in degrees

**Parameters**

| | |
|---|---|
| *in* | input stream |
| *pt* | angle to read value into |

**Returns**

The input stream

### 10.2.5 Member Data Documentation

#### 10.2.5.1 const Angle gazebo::math::Angle::HalfPi [static]

**math::Angle** (p. 137)(M_PI ∗ 0.5)

#### 10.2.5.2 const Angle gazebo::math::Angle::Pi [static]

math::Angle(M_PI)

#### 10.2.5.3 const Angle gazebo::math::Angle::TwoPi [static]

math::Angle(M_PI ∗ 2)

#### 10.2.5.4 const Angle gazebo::math::Angle::Zero [static]

math::Angle(0)

The documentation for this class was generated from the following file:

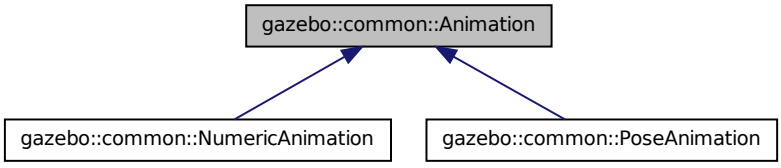
- **Angle.hh**

## 10.3 gazebo::common::Animation Class Reference

Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Animation:




### Public Member Functions

- **Animation** (const std::string &_name, double _length, bool _loop)

  *Constructor.*
- virtual ∼**Animation** ()

  *Destructor.*
- void **AddTime** (double _time)

  *Add time to the animation.*

- **KeyFrame** * **GetKeyFrame** (unsigned int _index) const

  *Get a key frame using an index value.*
- unsigned int **GetKeyFrameCount** () const

  *Return the number of key frames in the animation.*
- double **GetLength** () const

  *Return the duration of the animation.*
- double **GetTime** () const

  *Return the current time position.*
- void **SetLength** (double _len)

  *Set the duration of the animation.*
- void **SetTime** (double _time)

  *Set the current time position of the animation.*

### Protected Types

- typedef std::vector< **KeyFrame** * > **KeyFrame_V**

  *array of keyframe type alias*

### Protected Member Functions

- double **GetKeyFramesAtTime** (double _time, **KeyFrame** **_kf1, **KeyFrame** **_kf2, unsigned int &_firstKeyIndex) const

  *Get the two key frames that bound a time value.*

### Protected Attributes

- bool **build**

  *determines if the interpolation splines need building*
- **KeyFrame_V keyFrames**

  *array of key frames*
- double **length**

  *animation duration*
- bool **loop**

  *true if animation repeats*
- std::string **name**

  *animation name*
- double **timePos**

  *current time position*

#### 10.3.1 Detailed Description

Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.

### 10.3.2 Member Typedef Documentation

#### 10.3.2.1 typedef std::vector<KeyFrame*> gazebo::common::Animation::KeyFrame_V [protected]

array of keyframe type alias

### 10.3.3 Constructor & Destructor Documentation

#### 10.3.3.1 gazebo::common::Animation::Animation ( const std::string & *_name,* double *_length,* bool *_loop* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the animation, should be unique |
| in | *_length* | Duration of the animation in seconds |
| in | *_loop* | Set to true if the animation should repeat |

#### 10.3.3.2 virtual gazebo::common::Animation::~Animation ( ) [virtual]

Destructor.

### 10.3.4 Member Function Documentation

#### 10.3.4.1 void gazebo::common::Animation::AddTime ( double *_time* )

Add time to the animation.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The amount of time to add in seconds |

#### 10.3.4.2 KeyFrame* gazebo::common::Animation::GetKeyFrame ( unsigned int *_index* ) const

Get a key frame using an index value.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | The index of the key frame |

**Returns**

A pointer the keyframe, NULL if the _index is invalid

#### 10.3.4.3 unsigned int gazebo::common::Animation::GetKeyFrameCount ( ) const

Return the number of key frames in the animation.

**Returns**

The number of keyframes

#### 10.3.4.4 double gazebo::common::Animation::GetKeyFramesAtTime ( double *_time*, KeyFrame ** *_kf1*, KeyFrame ** *_kf2*, unsigned int & *_firstKeyIndex* ) const [protected]

Get the two key frames that bound a time value.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The time in seconds |
| out | *_kf1* | Lower bound keyframe that is returned |
| out | *_kf2* | Upper bound keyframe that is returned |
| out | *_firstKeyIndex* | Index of the lower bound key frame |

**Returns**

The time between the two keyframe

#### 10.3.4.5 double gazebo::common::Animation::GetLength ( ) const

Return the duration of the animation.

**Returns**

Duration of the animation in seconds

#### 10.3.4.6 double gazebo::common::Animation::GetTime ( ) const

Return the current time position.

**Returns**

The time position in seconds

#### 10.3.4.7 void gazebo::common::Animation::SetLength ( double *_len* )

Set the duration of the animation.

**Parameters**

| | | |
|---|---|---|
| in | *_len* | The length of the animation in seconds |

#### 10.3.4.8 void gazebo::common::Animation::SetTime ( double *_time* )

Set the current time position of the animation.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The time position in seconds |

### 10.3.5 Member Data Documentation

#### 10.3.5.1 bool gazebo::common::Animation::build [mutable], [protected]

determines if the interpolation splines need building

#### 10.3.5.2 KeyFrame_V gazebo::common::Animation::keyFrames [protected]

array of key frames

#### 10.3.5.3 double gazebo::common::Animation::length [protected]

animation duration

#### 10.3.5.4 bool gazebo::common::Animation::loop [protected]

true if animation repeats

#### 10.3.5.5 std::string gazebo::common::Animation::name [protected]

animation name

#### 10.3.5.6 double gazebo::common::Animation::timePos [protected]

current time position

The documentation for this class was generated from the following file:

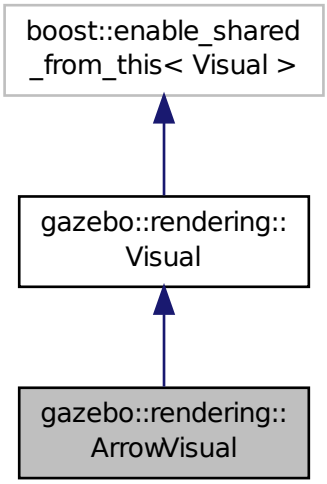
- **Animation.hh**

## 10.4 gazebo::rendering::ArrowVisual Class Reference

Basic arrow visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ArrowVisual:




## Public Member Functions

- **ArrowVisual** (const std::string &_name, **VisualPtr** _vis)

  *Constructor.*
- virtual ~**ArrowVisual** ()

  *Destructor.*
- virtual void **Load** ()

  *Load the visual with default parameters.*
- void **ShowRotation** ()

  *Show the rotation of the visual.*

## Additional Inherited Members

### 10.4.1 Detailed Description

Basic arrow visualization.

### 10.4.2 Constructor & Destructor Documentation

#### 10.4.2.1 gazebo::rendering::ArrowVisual::ArrowVisual ( const std::string & *_name,* VisualPtr *_vis* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the arrow visual |
| in | *_vis* | Pointer to the parent visual |

#### 10.4.2.2 virtual gazebo::rendering::ArrowVisual::~ArrowVisual ( ) [virtual]

Destructor.

### 10.4.3 Member Function Documentation

#### 10.4.3.1 virtual void gazebo::rendering::ArrowVisual::Load ( ) [virtual]

Load the visual with default parameters.

Reimplemented from **gazebo::rendering::Visual** (p. 1135).

#### 10.4.3.2 void gazebo::rendering::ArrowVisual::ShowRotation ( )

Show the rotation of the visual.

The documentation for this class was generated from the following file:

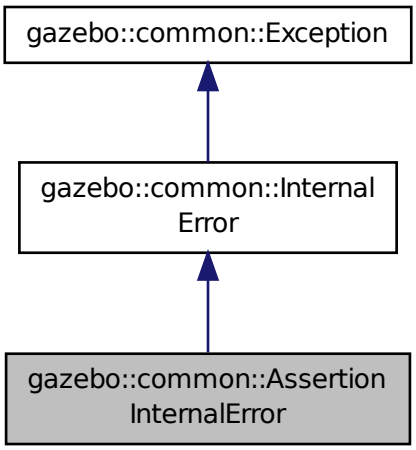
- **ArrowVisual.hh**

## 10.5 gazebo::common::AssertionInternalError Class Reference

Class for generating Exceptions which come from gazebo assertions.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::AssertionInternalError:




### Public Member Functions

- **AssertionInternalError** (const char ∗_file, int _line, const std::string &_expr, const std::string &_function, const std::string &_msg="")

  *Constructor for assertions.*
- virtual ∼**AssertionInternalError** ()

  *Destructor.*

## 10.5.1 Detailed Description

Class for generating Exceptions which come from gazebo assertions.

They include information about the assertion expression violated, function where problem appeared and assertion debug message.

## 10.5.2 Constructor & Destructor Documentation

### 10.5.2.1 gazebo::common::AssertionInternalError::AssertionInternalError ( const char ∗ *_file*, int *_line*, const std::string & *_expr*, const std::string & *_function*, const std::string & *_msg* = " " )

Constructor for assertions.

**Parameters**

| | | |
|---|---|---|
| in | *_file* | File name |
| in | *_line* | Line number where the error occurred |
| in | *_expr* | Assertion expression failed resulting in an internal error |

| | | |
|---|---|---|
| in | *_function* | Function where assertion failed |
| in | *_msg* | Function where assertion failed |

#### 10.5.2.2 virtual gazebo::common::AssertionInternalError::~AssertionInternalError ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Exception.hh**

## 10.6 gazebo::common::AudioDecoder Class Reference

An audio decoder based on FFMPEG.

`#include <common/common.hh>`

### Public Member Functions

- **AudioDecoder** ()

  *Constructor.*
- virtual ~**AudioDecoder** ()

  *Destructor.*
- bool **Decode** (uint8_t **_outBuffer, unsigned int *_outBufferSize)

  *Decode the loaded audio file.*
- std::string **GetFile** () const

  *Get the audio filename that was set.*
- int **GetSampleRate** ()

  *Get the sample rate from the latest decoded file.*
- bool **SetFile** (const std::string &_filename)

  *Set the file to decode.*

### 10.6.1 Detailed Description

An audio decoder based on FFMPEG.

### 10.6.2 Constructor & Destructor Documentation

#### 10.6.2.1 gazebo::common::AudioDecoder::AudioDecoder ( )

Constructor.

#### 10.6.2.2 virtual gazebo::common::AudioDecoder::~AudioDecoder ( ) [virtual]

Destructor.

### 10.6.3 Member Function Documentation

#### 10.6.3.1 bool gazebo::common::AudioDecoder::Decode ( uint8_t ** *_outBuffer,* unsigned int * *_outBufferSize* )

Decode the loaded audio file.

**See Also**

**AudioDecoder::SetFile** (p. 154)

**Parameters**

| | | |
|---|---|---|
| out | *_outBuffer* | Buffer that holds the decoded audio data. |
| out | *_outBufferSize* | Size of the _outBuffer. |

**Returns**

True if decoding was succesful.

#### 10.6.3.2 std::string gazebo::common::AudioDecoder::GetFile ( ) const

Get the audio filename that was set.

**Returns**

The name of the set audio file.

**See Also**

**AudioDecoder::SetFile** (p. 154)

#### 10.6.3.3 int gazebo::common::AudioDecoder::GetSampleRate ( )

Get the sample rate from the latest decoded file.

**Returns**

Integer sample rate, such as 44100.

#### 10.6.3.4 bool gazebo::common::AudioDecoder::SetFile ( const std::string & *_filename* )

Set the file to decode.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | Path to an audio file. |

**Returns**

True if the file was successfull opened.

The documentation for this class was generated from the following file:

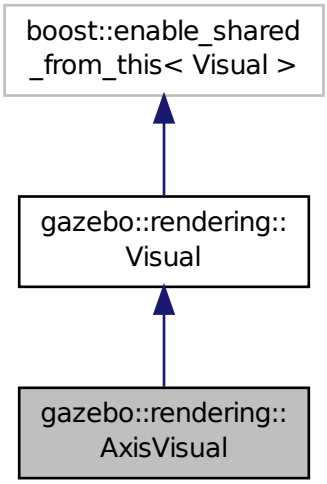
- **AudioDecoder.hh**

## 10.7 gazebo::rendering::AxisVisual Class Reference

Basic axis visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::AxisVisual:




### Public Member Functions

- **AxisVisual** (const std::string &_name, **VisualPtr** _vis)

  *Constructor.*
- virtual ~**AxisVisual** ()

  *Destructor.*
- virtual void **Load** ()

  *Load the axis visual.*
- void **ScaleXAxis** (const **math::Vector3** &_scale)

  *Scale the X axis.*
- void **ScaleYAxis** (const **math::Vector3** &_scale)

  *Scale the Y axis.*
- void **ScaleZAxis** (const **math::Vector3** &_scale)

  *Scale the Z axis.*
- void **SetAxisMaterial** (unsigned int _axis, const std::string &_material)

*Set the material used to render and axis.*

- void **ShowRotation** (unsigned int _axis)

  *Load the rotation tube.*

### Additional Inherited Members

### 10.7.1 Detailed Description

Basic axis visualization.

### 10.7.2 Constructor & Destructor Documentation

#### 10.7.2.1 gazebo::rendering::AxisVisual::AxisVisual ( const std::string & *_name,* VisualPtr *_vis* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the **AxisVisual** (p. 155) |
| in | *_vis* | Parent visual |

#### 10.7.2.2 virtual gazebo::rendering::AxisVisual::~AxisVisual ( ) [virtual]

Destructor.

### 10.7.3 Member Function Documentation

#### 10.7.3.1 virtual void gazebo::rendering::AxisVisual::Load ( ) [virtual]

Load the axis visual.

Reimplemented from **gazebo::rendering::Visual** (p. 1135).

#### 10.7.3.2 void gazebo::rendering::AxisVisual::ScaleXAxis ( const math::Vector3 & *_scale* )

Scale the X axis.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scaling factor |

#### 10.7.3.3 void gazebo::rendering::AxisVisual::ScaleYAxis ( const math::Vector3 & *_scale* )

Scale the Y axis.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scaling factor |

#### 10.7.3.4 void gazebo::rendering::AxisVisual::ScaleZAxis ( const math::Vector3 & _scale )

Scale the Z axis.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scaling factor |

#### 10.7.3.5 void gazebo::rendering::AxisVisual::SetAxisMaterial ( unsigned int _axis, const std::string & _material )

Set the material used to render and axis.

**Parameters**

| | | |
|---|---|---|
| in | *_axis* | The number of the axis (0, 1, 2 = x,y,z) |
| in | *_material* | The name of the material to apply to the axis |

#### 10.7.3.6 void gazebo::rendering::AxisVisual::ShowRotation ( unsigned int _axis )

Load the rotation tube.

The documentation for this class was generated from the following file:

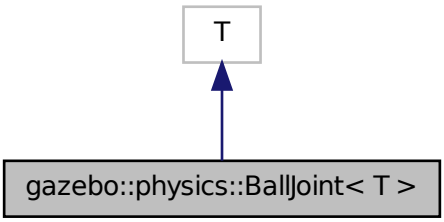
- **AxisVisual.hh**

## 10.8 gazebo::physics::BallJoint< T > Class Template Reference

**Base** (p. 159) class for a ball joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::BallJoint< T >:




### Public Member Functions

- **BallJoint** (**BasePtr** _parent)

*Constructor.*

- virtual ~**BallJoint** ()

  *Destructor.*
- virtual unsigned int **GetAngleCount** () const
- virtual **math::Angle GetHighStop** (int)
- virtual **math::Angle GetLowStop** (int)
- void **Load** (sdf::ElementPtr _sdf)

  *Template to ::Load the **BallJoint** (p. 157).*
- virtual void **SetAxis** (int, const **math::Vector3** &)
- virtual void **SetHighStop** (int, **math::Angle**)
- virtual void **SetLowStop** (int, **math::Angle**)

### 10.8.1 Detailed Description

**template<class T>class gazebo::physics::BallJoint< T >**

**Base** (p. 159) class for a ball joint.

Each physics engine should implement this class.

### 10.8.2 Constructor & Destructor Documentation

#### 10.8.2.1 template<class T> gazebo::physics::BallJoint< T >::BallJoint ( BasePtr *_parent* ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Pointer to the parent link. |

#### 10.8.2.2 template<class T> virtual gazebo::physics::BallJoint< T >::~BallJoint ( ) [inline], [virtual]

Destructor.

### 10.8.3 Member Function Documentation

#### 10.8.3.1 template<class T> virtual unsigned int gazebo::physics::BallJoint< T >::GetAngleCount ( ) const [inline], [virtual]

#### 10.8.3.2 template<class T> virtual math::Angle gazebo::physics::BallJoint< T >::GetHighStop ( int ) [inline], [virtual]

#### 10.8.3.3 template<class T> virtual math::Angle gazebo::physics::BallJoint< T >::GetLowStop ( int ) [inline], [virtual]

#### 10.8.3.4 template<class T> void gazebo::physics::BallJoint< T >::Load ( sdf::ElementPtr *_sdf* ) [inline]

Template to ::Load the **BallJoint** (p. 157).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF to load the joint from. |

#### 10.8.3.5 template<class T> virtual void gazebo::physics::BallJoint< T >::SetAxis ( int , const math::Vector3 & )

[inline], [virtual]

#### 10.8.3.6 template<class T> virtual void gazebo::physics::BallJoint< T >::SetHighStop ( int , math::Angle )

[inline], [virtual]

#### 10.8.3.7 template<class T> virtual void gazebo::physics::BallJoint< T >::SetLowStop ( int , math::Angle )

[inline], [virtual]

The documentation for this class was generated from the following file:

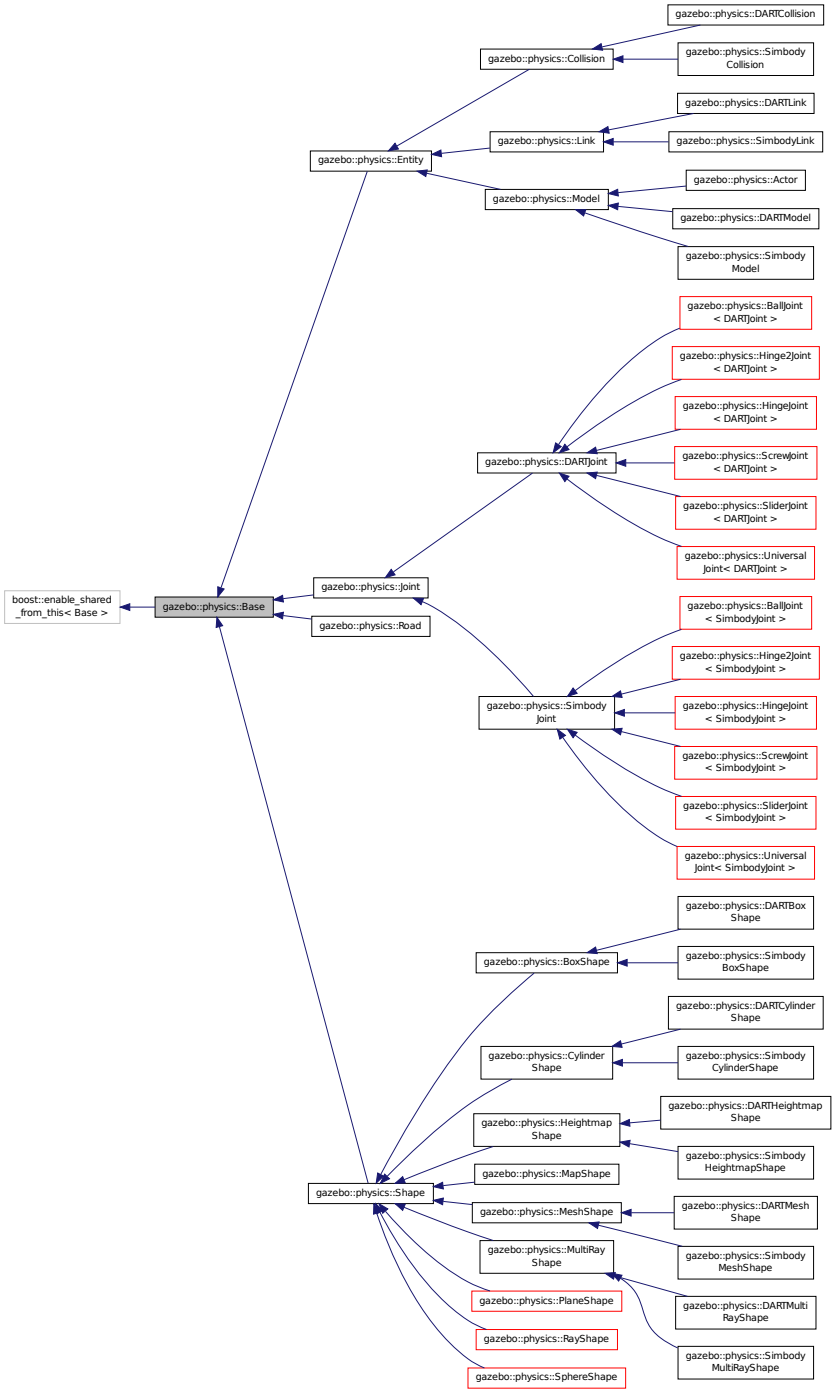
- **BallJoint.hh**

## 10.9 gazebo::physics::Base Class Reference

**Base** (p. 159) class for most physics classes.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Base:



## Public Types

- enum **EntityType** {
  **BASE** = 0x00000000, **ENTITY** = 0x00000001, **MODEL** = 0x00000002, **LINK** = 0x00000004,
  **COLLISION** = 0x00000008, **ACTOR** = 0x00000016, **LIGHT** = 0x00000010, **VISUAL** = 0x00000020,
  **JOINT** = 0x00000040, **BALL_JOINT** = 0x00000080, **HINGE2_JOINT** = 0x00000100, **HINGE_JOINT** = 0x00000200,
  **SLIDER_JOINT** = 0x00000400, **SCREW_JOINT** = 0x00000800, **UNIVERSAL_JOINT** = 0x00001000, **SHAPE** = 0x00002000,
  **BOX_SHAPE** = 0x00004000, **CYLINDER_SHAPE** = 0x00008000, **HEIGHTMAP_SHAPE** = 0x00010000, **MAP-_SHAPE** = 0x00020000,
  **MULTIRAY_SHAPE** = 0x00040000, **RAY_SHAPE** = 0x00080000, **PLANE_SHAPE** = 0x00100000, **SPHERE_-SHAPE** = 0x00200000,
  **MESH_SHAPE** = 0x00400000, **SENSOR_COLLISION** = 0x00800000 }

  *Unique identifiers for all entity types.*

## Public Member Functions

- **Base** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**Base** ()

  *Destructor.*
- void **AddChild** (**BasePtr** _child)

  *Add a child to this entity.*
- void **AddType** (**EntityType** _type)

  *Add a type specifier.*
- virtual void **Fini** ()

  *Finialize the object.*
- **BasePtr GetByName** (const std::string &_name)

  *Get by name.*
- **BasePtr GetChild** (unsigned int _i) const

  *Get a child by index.*
- **BasePtr GetChild** (const std::string &_name)

  *Get a child by name.*
- unsigned int **GetChildCount** () const

  *Get the number of children.*
- uint32_t **GetId** () const

  *Return the ID of this entity.*
- std::string **GetName** () const

  *Return the name of the entity.*
- **BasePtr GetParent** () const

  *Get the parent.*
- int **GetParentId** () const

  *Return the ID of the parent.*
- bool **GetSaveable** () const

  *Get whether the object should be "saved", when the user selects to save the world to xml.*
- std::string **GetScopedName** () const

  *Return the name of this entity with the model scope world::model1::...::modelN::entityName.*
- virtual const sdf::ElementPtr **GetSDF** ()

*Get the SDF values for the object.*

- unsigned int **GetType** () const

  *Get the full type definition.*

- const **WorldPtr** & **GetWorld** () const

  *Get the **World** (p. 1157) this object is in.*

- bool **HasType** (const **EntityType** &_t) const

  *Returns true if this object's type definition has the given type.*

- virtual void **Init** ()

  *Initialize the object.*

- bool **IsSelected** () const

  *True if the entity is selected by the user.*

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load.*

- bool **operator==** (const **Base** &_ent) const

  *Returns true if the entities are the same.*

- void **Print** (const std::string &_prefix)

  *Print this object to screen via gzmsg.*

- virtual void **RemoveChild** (unsigned int _id)

  *Remove a child from this entity.*

- void **RemoveChild** (const std::string &_name)

  *Remove a child by name.*

- void **RemoveChildren** ()

  *Remove all children.*

- virtual void **Reset** ()

  *Reset the object.*

- virtual void **Reset** (**Base::EntityType** _resetType)

  *Calls recursive Reset on one of the **Base::EntityType** (p. 163)'s.*

- virtual void **SetName** (const std::string &_name)

  *Set the name of the entity.*

- void **SetParent** (**BasePtr** _parent)

  *Set the parent.*

- void **SetSaveable** (bool _v)

  *Set whether the object should be "saved", when the user selects to save the world to xml.*

- virtual bool **SetSelected** (bool _show)

  *Set whether this entity has been selected by the user through the gui.*

- void **SetWorld** (const **WorldPtr** &_newWorld)

  *Set the world this object belongs to.*

- virtual void **Update** ()

  *Update the object.*

- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  *Update the parameters using new sdf values.*

## Protected Member Functions

- void **ComputeScopedName** ()

  *Compute the scoped name of this object based on its parents.*

### Protected Attributes

- **Base_V children**

  *Children of this entity.*
- Base_V::iterator **childrenEnd**

  *End of the children vector.*
- **BasePtr parent**

  *Parent of this entity.*
- sdf::ElementPtr **sdf**

  *The SDF values for this object.*
- **WorldPtr world**

  *Pointer to the world.*

## 10.9.1 Detailed Description

**Base** (p. 159) class for most physics classes.

## 10.9.2 Member Enumeration Documentation

### 10.9.2.1 enum gazebo::physics::Base::EntityType

Unique identifiers for all entity types.

**Enumerator**

***BASE*** **Base** (p. 159) type.

***ENTITY*** **Entity** (p. 379) type.

***MODEL*** **Model** (p. 624) type.

***LINK*** **Link** (p. 542) type.

***COLLISION*** **Collision** (p. 220) type.

***ACTOR*** **Actor** (p. 131) type.

***LIGHT*** Light type.

***VISUAL*** Visual type.

***JOINT*** **Joint** (p. 496) type.

***BALL_JOINT*** **BallJoint** (p. 157) type.

***HINGE2_JOINT*** Hing2Joint type.

***HINGE_JOINT*** **HingeJoint** (p. 472) type.

***SLIDER_JOINT*** **SliderJoint** (p. 973) type.

***SCREW_JOINT*** **ScrewJoint** (p. 832) type.

***UNIVERSAL_JOINT*** **UniversalJoint** (p. 1064) type.

***SHAPE*** **Shape** (p. 861) type.

***BOX_SHAPE*** **BoxShape** (p. 176) type.

***CYLINDER_SHAPE*** **CylinderShape** (p. 275) type.

***HEIGHTMAP_SHAPE*** **HeightmapShape** (p. 465) type.

***MAP_SHAPE*** **MapShape** (p. 579) type.

***MULTIRAY_SHAPE*** **MultiRayShape** (p. 664) type.

***RAY_SHAPE*** **RayShape** (p. 786) type.

***PLANE_SHAPE*** **PlaneShape** (p. 728) type.

***SPHERE_SHAPE*** **SphereShape** (p. 986) type.

***MESH_SHAPE*** **MeshShape** (p. 621) type.

***SENSOR_COLLISION*** Indicates a collision shape used for sensing.

### 10.9.3 Constructor & Destructor Documentation

#### 10.9.3.1 gazebo::physics::Base::Base ( BasePtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of this object |

#### 10.9.3.2 virtual gazebo::physics::Base::~Base ( ) [virtual]

Destructor.

### 10.9.4 Member Function Documentation

#### 10.9.4.1 void gazebo::physics::Base::AddChild ( BasePtr *_child* )

Add a child to this entity.

**Parameters**

| | | |
|---|---|---|
| in | *_child* | Child entity. |

#### 10.9.4.2 void gazebo::physics::Base::AddType ( EntityType *_type* )

Add a type specifier.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | New type to append to this objects type definition. |

#### 10.9.4.3 void gazebo::physics::Base::ComputeScopedName ( ) [protected]

Compute the scoped name of this object based on its parents.

**See Also**

**Base::GetScopedName** (p. 167)

#### 10.9.4.4 virtual void gazebo::physics::Base::Fini ( ) [virtual]

Finialize the object.

Reimplemented in **gazebo::physics::Actor** (p. 134), **gazebo::physics::Link** (p. 550), **gazebo::physics::Model** (p. 629), **gazebo::physics::Entity** (p. 382), **gazebo::physics::DARTModel** (p. 328), **gazebo::physics::Collision** (p. 223), **gazebo::physics::DARTLink** (p. 318), **gazebo::physics::SimbodyLink** (p. 905), and **gazebo::physics::-DARTCollision** (p. 288).

#### 10.9.4.5 BasePtr gazebo::physics::Base::GetByName ( const std::string & *_name* )

Get by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Get a child (or self) object by name |

**Returns**

A pointer to the object, NULL if not found

#### 10.9.4.6 BasePtr gazebo::physics::Base::GetChild ( unsigned int *_i* ) const

Get a child by index.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | Index of the child to retreive. |

**Returns**

A pointer to the object, NULL if the index is invalid.

#### 10.9.4.7 BasePtr gazebo::physics::Base::GetChild ( const std::string & *_name* )

Get a child by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the child. |

**Returns**

A pointer to the object, NULL if not found

#### 10.9.4.8 unsigned int gazebo::physics::Base::GetChildCount ( ) const

Get the number of children.

**Returns**

The number of children.

#### 10.9.4.9 uint32_t gazebo::physics::Base::GetId ( ) const

Return the ID of this entity.

This id is unique.

**Returns**

Integer ID.

#### 10.9.4.10 std::string gazebo::physics::Base::GetName ( ) const

Return the name of the entity.

**Returns**

Name of the entity.

#### 10.9.4.11 BasePtr gazebo::physics::Base::GetParent ( ) const

Get the parent.

**Returns**

Pointer to the parent entity.

#### 10.9.4.12 int gazebo::physics::Base::GetParentId ( ) const

Return the ID of the parent.

**Returns**

Integer ID.

#### 10.9.4.13 bool gazebo::physics::Base::GetSaveable ( ) const

Get whether the object should be "saved", when the user selects to save the world to xml.

**Returns**

True if the object is saveable.

#### 10.9.4.14 std::string gazebo::physics::Base::GetScopedName ( ) const

Return the name of this entity with the model scope world::model1::...::modelN::entityName.

**Returns**

The scoped name.

#### 10.9.4.15 virtual const sdf::ElementPtr gazebo::physics::Base::GetSDF ( ) [virtual]

Get the SDF values for the object.

**Returns**

The SDF values for the object.

Reimplemented in **gazebo::physics::Actor** (p. 134), and **gazebo::physics::Model** (p. 632).

#### 10.9.4.16 unsigned int gazebo::physics::Base::GetType ( ) const

Get the full type definition.

**Returns**

The full type definition.

#### 10.9.4.17 const WorldPtr& gazebo::physics::Base::GetWorld ( ) const

Get the **World** (p. 1157) this object is in.

**Returns**

The **World** (p. 1157) this object is part of.

#### 10.9.4.18 bool gazebo::physics::Base::HasType ( const EntityType & _t ) const

Returns true if this object's type definition has the given type.

**Parameters**

| | | |
|---|---|---|
| in | _t | Type to check. |

**Returns**

True if this object's type definition has the.

#### 10.9.4.19 virtual void gazebo::physics::Base::Init ( ) [inline], [virtual]

Initialize the object.

Reimplemented in **gazebo::physics::Joint** (p. 510), **gazebo::physics::RayShape** (p. 789), **gazebo::physics::Link** (p. 556), **gazebo::physics::Actor** (p. 134), **gazebo::physics::Model** (p. 633), **gazebo::physics::MapShape** (p. 582), **gazebo::physics::HingeJoint< DARTJoint >** (p. 474), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 474), **gazebo::physics::Collision** (p. 226), **gazebo::physics::HeightmapShape** (p. 469), **gazebo::physics::MeshShape** (p. 623), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyScrewJoint** (p. 934), **gazebo::physics::DARTLink** (p. 321), **gazebo::physics::MultiRayShape** (p. 671), **gazebo::physics::PlaneShape** (p. 731), **gazebo::physics::SimbodyBallJoint** (p. 869), **gazebo::physics::SimbodyLink** (p. 907), **gazebo::physics::DARTModel** (p. 328), **gazebo::physics::Road** (p. 805), **gazebo::physics::Shape** (p. 864), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::DARTJoint** (p. 311), **gazebo::physics::SphereShape** (p. 988), **gazebo::physics::BoxShape** (p. 178), **gazebo::physics::CylinderShape** (p. 278), **gazebo::physics::DARTCollision** (p. 289), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::SimbodyHeightmapShape** (p. 879), **gazebo::physics::SimbodyModel** (p. 913), **gazebo::physics::SimbodyMeshShape** (p. 911), **gazebo::physics::DARTBallJoint** (p. 283), **gazebo::physics::DARTHeightmapShape** (p. 294), **gazebo::physics::DARTScrewJoint** (p. 343), **gazebo::physics::DARTSliderJoint** (p. 348), **gazebo::physics::DARTUniversalJoint** (p. 356), and **gazebo::physics::DARTMeshShape** (p. 325).

#### 10.9.4.20 bool gazebo::physics::Base::IsSelected ( ) const

True if the entity is selected by the user.

**Returns**

True if the entity is selected.

#### 10.9.4.21 virtual void gazebo::physics::Base::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load.

**Parameters**

| | | |
|---|---|---|
| in | *node* | Pointer to an SDF parameters |

Reimplemented in **gazebo::physics::Joint** (p. 510), **gazebo::physics::SimbodySliderJoint** (p. 940), **gazebo::physics::Link** (p. 556), **gazebo::physics::Actor** (p. 134), **gazebo::physics::Entity** (p. 386), **gazebo::physics::Model** (p. 633), **gazebo::physics::MapShape** (p. 582), **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 472), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 472), **gazebo::physics::Collision** (p. 227), **gazebo::physics::HeightmapShape** (p. 470), **gazebo::physics::BallJoint< DARTJoint >** (p. 158), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 158), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1065), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1065), **gazebo::physics::HingeJoint< DARTJoint >** (p. 474), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 474), **gazebo::physics::SliderJoint< DARTJoint >** (p. 975), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 975), **gazebo::physics::SimbodyCollision** (p. 874), **gazebo::physics::DARTLink** (p. 321), **gazebo::physics::SimbodyHingeJoint** (p. 889), **gazebo::physics::SimbodyLink** (p. 907), **gazebo::physics::DARTModel** (p. 328), **gazebo::physics::Road** (p. 805), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::SimbodyJoint** (p. 897), **gazebo::physics::SimbodyScrewJoint** (p. 934), **gazebo::physics::DARTJoint** (p. 311), **gazebo::physics::DARTCollision** (p. 289), **gazebo::physics::SimbodyBallJoint** (p. 869), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::SimbodyModel** (p. 913), **gazebo::physics::SimbodyMeshShape** (p. 911), **gazebo::physics::DARTBallJoint** (p. 283), **gazebo::physics::DARTScrewJoint** (p. 343), **gazebo::physics::DARTSliderJoint** (p. 348), **gazebo::physics::DARTUniversalJoint** (p. 356), and **gazebo::physics::DARTMeshShape** (p. 325).

#### 10.9.4.22 bool gazebo::physics::Base::operator== ( const Base & _ent ) const

Returns true if the entities are the same.

Checks only the name.

**Parameters**

| | | |
|---|---|---|
| in | *_ent* | **Base** (p. 159) object to compare with. |

**Returns**

True if the entities are the same.

#### 10.9.4.23 void gazebo::physics::Base::Print ( const std::string & _prefix )

Print this object to screen via gzmsg.

**Parameters**

| | | |
|---|---|---|
| in | *_prefix* | Usually a set of spaces. |

#### 10.9.4.24 virtual void gazebo::physics::Base::RemoveChild ( unsigned int _id ) [virtual]

Remove a child from this entity.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the child to remove. |

#### 10.9.4.25 void gazebo::physics::Base::RemoveChild ( const std::string & _name )

Remove a child by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the child. |

#### 10.9.4.26 void gazebo::physics::Base::RemoveChildren ( )

Remove all children.

#### 10.9.4.27 virtual void gazebo::physics::Base::Reset ( ) [virtual]

Reset the object.

Reimplemented in **gazebo::physics::Joint** (p. 511), **gazebo::physics::Model** (p. 634), **gazebo::physics::Link** (p. 557), **gazebo::physics::Entity** (p. 387), **gazebo::physics::DARTJoint** (p. 311), and **gazebo::physics::SimbodyJoint** (p. 897).

#### 10.9.4.28 virtual void gazebo::physics::Base::Reset ( Base::EntityType _resetType ) [virtual]

Calls recursive Reset on one of the **Base::EntityType** (p. 163)'s.

**Parameters**

| | | |
|---|---|---|
| in | *_resetType* | The type of reset operation |

#### 10.9.4.29 virtual void gazebo::physics::Base::SetName ( const std::string & _name ) [virtual]

Set the name of the entity.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | New name. |

Reimplemented in **gazebo::physics::Entity** (p. 388).

#### 10.9.4.30 void gazebo::physics::Base::SetParent ( BasePtr _parent )

Set the parent.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent object. |

#### 10.9.4.31 void gazebo::physics::Base::SetSaveable ( bool _v )

Set whether the object should be "saved", when the user selects to save the world to xml.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | Set to True if the object should be saved. |

#### 10.9.4.32 virtual bool gazebo::physics::Base::SetSelected ( bool _show ) [virtual]

Set whether this entity has been selected by the user through the gui.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to set this entity as selected. |

Reimplemented in **gazebo::physics::Link** (p. 561).

#### 10.9.4.33 void gazebo::physics::Base::SetWorld ( const WorldPtr & _newWorld )

Set the world this object belongs to.

This will also set the world for all children.

**Parameters**

| | | |
|---|---|---|
| in | *_newWorld* | The new **World** (p. 1157) this object is part of. |

#### 10.9.4.34 virtual void gazebo::physics::Base::Update ( ) [inline], [virtual]

Update the object.

Reimplemented in **gazebo::physics::Joint** (p. 514), **gazebo::physics::MultiRayShape** (p. 671), **gazebo::physics::RayShape** (p. 791), **gazebo::physics::Actor** (p. 135), **gazebo::physics::Model** (p. 637), **gazebo::physics::MapShape** (p. 583), **gazebo::physics::DARTModel** (p. 328), **gazebo::physics::DARTRayShape** (p. 75), **gazebo::physics::MeshShape** (p. 624), **gazebo::physics::SimbodyRayShape** (p. 929), and **gazebo::physics::DARTMeshShape** (p. 326).

#### 10.9.4.35 virtual void gazebo::physics::Base::UpdateParameters ( sdf::ElementPtr *_sdf* ) [virtual]

Update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Update the object's parameters based on SDF values. |

Reimplemented in **gazebo::physics::Joint** (p. 514), **gazebo::physics::Actor** (p. 135), **gazebo::physics::Link** (p. 562), **gazebo::physics::Model** (p. 638), **gazebo::physics::Entity** (p. 389), and **gazebo::physics::Collision** (p. 229).

### 10.9.5 Member Data Documentation

#### 10.9.5.1 Base_V gazebo::physics::Base::children [protected]

Children of this entity.

#### 10.9.5.2 Base_V::iterator gazebo::physics::Base::childrenEnd [protected]

End of the children vector.

#### 10.9.5.3 BasePtr gazebo::physics::Base::parent [protected]

Parent of this entity.

#### 10.9.5.4 sdf::ElementPtr gazebo::physics::Base::sdf [protected]

The SDF values for this object.

#### 10.9.5.5 WorldPtr gazebo::physics::Base::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **Base.hh**

## 10.10 gazebo::math::Box Class Reference

Mathematical representation of a box and related functions.

`#include <math/gzmath.hh>`

### Public Member Functions

- **Box** ()
  *Default constructor.*
- **Box** (const **Vector3** &_min, const **Vector3** &_max)
  *Constructor.*
- **Box** (const **Box** &_b)
  *Copy Constructor.*
- virtual ~**Box** ()
  *Destructor.*
- **math::Vector3 GetCenter** () const
  *Get the box center.*
- **math::Vector3 GetSize** () const
  *Get the size of the box.*
- double **GetXLength** () const
  *Get the length along the x dimension.*
- double **GetYLength** () const
  *Get the length along the y dimension.*
- double **GetZLength** () const
  *Get the length along the z dimension.*
- void **Merge** (const **Box** &_box)
  *Merge a box with this box.*
- **Box operator+** (const **Box** &_b) const
  *Addition operator.*
- const **Box** & **operator+=** (const **Box** &_b)
  *Addition set operator.*
- **Box operator-** (const **Vector3** &_v)
  *Subtract a vector from the min and max values.*
- **Box** & **operator=** (const **Box** &_b)
  *Assignment operator.*
- bool **operator==** (const **Box** &_b)
  *Equality test operatoer.*

### Public Attributes

- **Vector3 max**
  *Maximum corner of the box.*
- **Vector3 min**
  *Minimum corner of the box.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Box** &_b)

  *Output operator.*

### 10.10.1 Detailed Description

Mathematical representation of a box and related functions.

### 10.10.2 Constructor & Destructor Documentation

#### 10.10.2.1 gazebo::math::Box::Box ( )

Default constructor.

#### 10.10.2.2 gazebo::math::Box::Box ( const Vector3 & *_min*, const Vector3 & *_max* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_min* | Minimum corner of the box |
| in | *_max* | Maximum corner of the box |

#### 10.10.2.3 gazebo::math::Box::Box ( const Box & *_b* )

Copy Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_b* | **Box** (p. 172) to copy |

#### 10.10.2.4 virtual gazebo::math::Box::~Box ( ) [virtual]

Destructor.

### 10.10.3 Member Function Documentation

#### 10.10.3.1 math::Vector3 gazebo::math::Box::GetCenter ( ) const

Get the box center.

**Returns**

The center position of the box

#### 10.10.3.2 math::Vector3 gazebo::math::Box::GetSize ( ) const

Get the size of the box.

**Returns**

Size of the box

#### 10.10.3.3 double gazebo::math::Box::GetXLength ( ) const

Get the length along the x dimension.

**Returns**

Double value of the length in the x dimension

#### 10.10.3.4 double gazebo::math::Box::GetYLength ( ) const

Get the length along the y dimension.

**Returns**

Double value of the length in the y dimension

#### 10.10.3.5 double gazebo::math::Box::GetZLength ( ) const

Get the length along the z dimension.

**Returns**

Double value of the length in the z dimension

#### 10.10.3.6 void gazebo::math::Box::Merge ( const Box & _box )

Merge a box with this box.

**Parameters**

| | | |
|---|---|---|
| in | *_box* | **Box** (p. 172) to add to this box |

#### 10.10.3.7 Box gazebo::math::Box::operator+ ( const Box & _b ) const

Addition operator.

result = this + _b

**Parameters**

| | | |
|---|---|---|
| in | *_b* | **Box** (p. 172) to add |

**Returns**

The new box

#### 10.10.3.8 const Box& gazebo::math::Box::operator+= ( const Box & _b )

Addition set operator.

this = this + _b

**Parameters**

| | | |
|---|---|---|
| in | *_b* | **Box** (p. 172) to add |

**Returns**

This new box

#### 10.10.3.9 Box gazebo::math::Box::operator- ( const Vector3 & _v )

Subtract a vector from the min and max values.

**Parameters**

| | |
|---|---|
| *_v* | The vector to use during subtraction |

**Returns**

The new box

#### 10.10.3.10 Box& gazebo::math::Box::operator= ( const Box & _b )

Assignment operator.

Set this box to the parameter

**Parameters**

| | | |
|---|---|---|
| in | *_b* | **Box** (p. 172) to copy |

**Returns**

The new box.

#### 10.10.3.11 bool gazebo::math::Box::operator== ( const Box & _b )

Equality test operatoer.

**Parameters**

| | | |
|---|---|---|
| in | *_b* | **Box** (p. 172) to test |

**Returns**

True if equal

### 10.10.4 Friends And Related Function Documentation

#### 10.10.4.1 std::ostream& operator<< ( std::ostream & *_out,* const gazebo::math::Box & *_b* ) [friend]

Output operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | Output stream |
| in | *_b* | **Box** (p. 172) to output to the stream |

**Returns**

The stream

### 10.10.5 Member Data Documentation

#### 10.10.5.1 Vector3 gazebo::math::Box::max

Maximum corner of the box.

#### 10.10.5.2 Vector3 gazebo::math::Box::min

Minimum corner of the box.

The documentation for this class was generated from the following file:

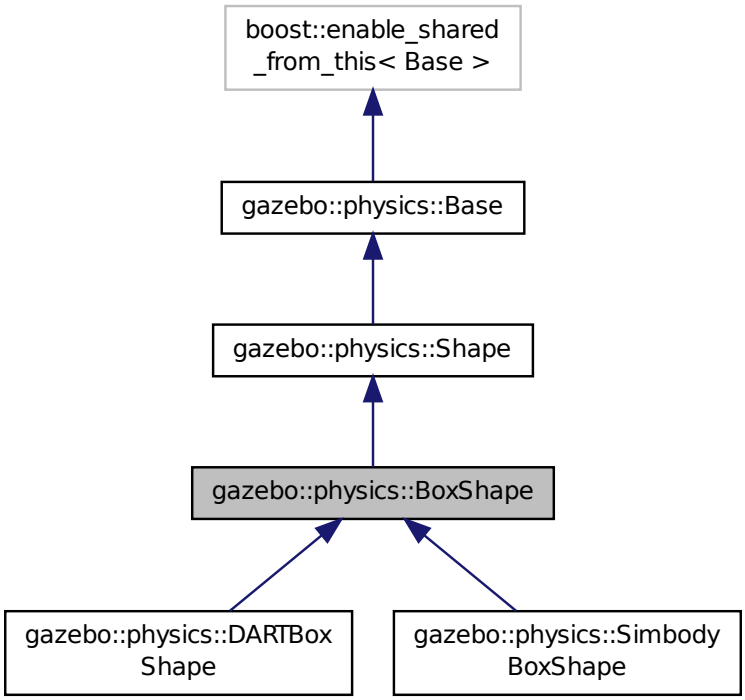
- **Box.hh**

## 10.11 gazebo::physics::BoxShape Class Reference

Box geometry primitive.

`#include <physics/physcs.hh>`

Inheritance diagram for gazebo::physics::BoxShape:




## Public Member Functions

- **BoxShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**BoxShape** ()

  *Destructor.*
- void **FillMsg** (msgs::Geometry &_msg)

  *Fill in the values for a geomertry message.*
- **math::Vector3 GetSize** () const

  *Get the size of the box.*
- virtual void **Init** ()

  *Initialize the box.*
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *Process a geometry message.*
- virtual void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the box.*
- virtual void **SetSize** (const **math::Vector3** &_size)

  *Set the size of the box.*

### Additional Inherited Members

### 10.11.1 Detailed Description

Box geometry primitive.

### 10.11.2 Constructor & Destructor Documentation

#### 10.11.2.1 gazebo::physics::BoxShape::BoxShape ( CollisionPtr _parent ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent **Collision** (p. 220). |

#### 10.11.2.2 virtual gazebo::physics::BoxShape::~BoxShape ( ) [virtual]

Destructor.

### 10.11.3 Member Function Documentation

#### 10.11.3.1 void gazebo::physics::BoxShape::FillMsg ( msgs::Geometry & _msg ) [virtual]

Fill in the values for a geomertry message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | The geometry message to fill. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.11.3.2 math::Vector3 gazebo::physics::BoxShape::GetSize ( ) const

Get the size of the box.

**Returns**

The size of each side of the box.

#### 10.11.3.3 virtual void gazebo::physics::BoxShape::Init ( ) [virtual]

Initialize the box.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.11.3.4 virtual void gazebo::physics::BoxShape::ProcessMsg ( const msgs::Geometry & _msg ) [virtual]

Process a geometry message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | The message to set values from. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.11.3.5 virtual void gazebo::physics::BoxShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set the scale of the box.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale of the box. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.11.3.6 virtual void gazebo::physics::BoxShape::SetSize ( const math::Vector3 & *_size* ) [virtual]

Set the size of the box.

**Parameters**

| | | |
|---|---|---|
| in | *_size* | Size of each side of the box. |

Reimplemented in **gazebo::physics::DARTBoxShape** (p. 286), and **gazebo::physics::SimbodyBoxShape** (p. 872).

Referenced by gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

The documentation for this class was generated from the following file:

- **BoxShape.hh**

## 10.12 gazebo::common::BVHLoader Class Reference

Handles loading BVH animation files.

`#include <common/common.hh>`

### Public Member Functions

- **BVHLoader** ()

  *Constructor.*
- ∼**BVHLoader** ()

  *Desutrctor.*
- **Skeleton** ∗ **Load** (const std::string &_filename, double _scale)

  *Load a BVH file.*

### 10.12.1 Detailed Description

Handles loading BVH animation files.

### 10.12.2 Constructor & Destructor Documentation

#### 10.12.2.1 gazebo::common::BVHLoader::BVHLoader ( )

Constructor.

#### 10.12.2.2 gazebo::common::BVHLoader::~BVHLoader ( )

Desutrctor.

### 10.12.3 Member Function Documentation

#### 10.12.3.1 Skeleton* gazebo::common::BVHLoader::Load ( const std::string & *_filename*, double *_scale* )

Load a BVH file.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | BVH file to load |
| in | *_scale* | Scaling factor to apply to the skeleton |

**Returns**

A pointer to a new **Skeleton** (p. 953)

The documentation for this class was generated from the following file:

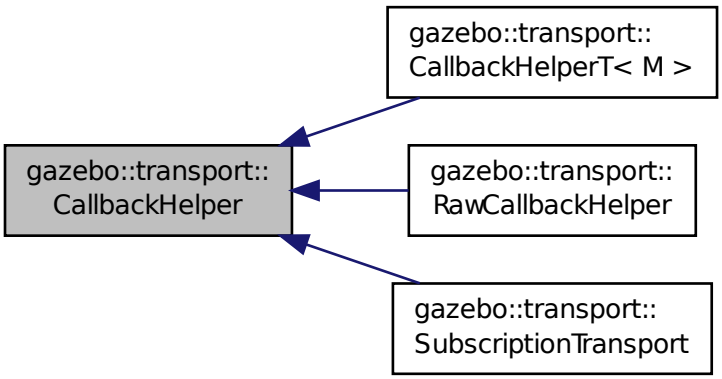
- **BVHLoader.hh**

## 10.13 gazebo::transport::CallbackHelper Class Reference

A helper class to handle callbacks when messages arrive.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::CallbackHelper:




### Public Member Functions

- **CallbackHelper** (bool _latching=false)

  *Constructor.*
- virtual ~**CallbackHelper** ()

  *Destructor.*
- unsigned int **GetId** () const

  *Get the unique ID of this callback.*
- bool **GetLatching** () const

  *Is the callback latching?*
- virtual std::string **GetMsgType** () const

  *Get the typename of the message that is handled.*
- virtual bool **HandleData** (const std::string &_newdata, boost::function< void(uint32_t)> _cb, uint32_t _id)=0

  *Process new incoming data.*
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)=0

  *Process new incoming message.*
- virtual bool **IsLocal** () const =0

  *Is the callback local?*

### Protected Attributes

- bool **latching**

  *True means that the callback helper will get the last published message on the topic.*

### 10.13.1 Detailed Description

A helper class to handle callbacks when messages arrive.

### 10.13.2 Constructor & Destructor Documentation

#### 10.13.2.1 gazebo::transport::CallbackHelper::CallbackHelper ( bool *_latching* = `false` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_latching* | Set to true to make the callback helper latching. |

#### 10.13.2.2 virtual gazebo::transport::CallbackHelper::~CallbackHelper ( ) [virtual]

Destructor.

### 10.13.3 Member Function Documentation

#### 10.13.3.1 unsigned int gazebo::transport::CallbackHelper::GetId ( ) const

Get the unique ID of this callback.

**Returns**

The unique ID of this callback.

#### 10.13.3.2 bool gazebo::transport::CallbackHelper::GetLatching ( ) const

Is the callback latching?

**Returns**

true if the callback is latching, false otherwise

#### 10.13.3.3 virtual std::string gazebo::transport::CallbackHelper::GetMsgType ( ) const [virtual]

Get the typename of the message that is handled.

**Returns**

String representation of the message type

Reimplemented in **gazebo::transport::RawCallbackHelper** (p. 778), and **gazebo::transport::CallbackHelperT< M >** (p. 185).

#### 10.13.3.4 virtual bool gazebo::transport::CallbackHelper::HandleData ( const std::string & *_newdata,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id* ) [pure virtual]

Process new incoming data.

**Parameters**

| | | |
|---|---|---|
| in | *_newdata* | Incoming data to be processed |

**Returns**

true if successfully processed; false otherwise

**Parameters**

| | | |
|---|---|---|
| in | *_cb* | If non-null, callback to be invoked which signals that transmission is complete. |
| in | *_id* | ID associated with the message data. |

Implemented in **gazebo::transport::RawCallbackHelper** (p. 778), **gazebo::transport::CallbackHelperT< M >** (p. 185), and **gazebo::transport::SubscriptionTransport** (p. 1019).

#### 10.13.3.5 virtual bool gazebo::transport::CallbackHelper::HandleMessage ( MessagePtr *_newMsg* ) [pure virtual]

Process new incoming message.

**Parameters**

| | | |
|---|---|---|
| in | *_newMsg* | Incoming message to be processed |

**Returns**

true if successfully processed; false otherwise

Implemented in **gazebo::transport::RawCallbackHelper** (p. 778), **gazebo::transport::CallbackHelperT< M >** (p. 185), and **gazebo::transport::SubscriptionTransport** (p. 1019).

#### 10.13.3.6 virtual bool gazebo::transport::CallbackHelper::IsLocal ( ) const [pure virtual]

Is the callback local?

**Returns**

true if the callback is local, false if the callback is tied to a remote connection

Implemented in **gazebo::transport::RawCallbackHelper** (p. 778), **gazebo::transport::CallbackHelperT< M >** (p. 186), and **gazebo::transport::SubscriptionTransport** (p. 1020).

### 10.13.4 Member Data Documentation

#### 10.13.4.1 bool gazebo::transport::CallbackHelper::latching [protected]

True means that the callback helper will get the last published message on the topic.

The documentation for this class was generated from the following file:

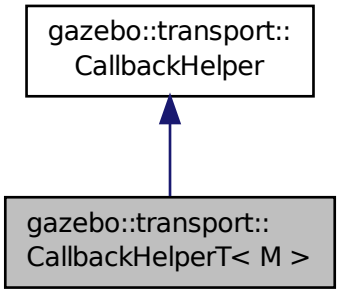
- **CallbackHelper.hh**

## 10.14 gazebo::transport::CallbackHelperT< M > Class Template Reference

Callback helper Template.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::CallbackHelperT< M >:




### Public Member Functions

- **CallbackHelperT** (const boost::function< void(const boost::shared_ptr< M const > &)> &_cb, bool _latching=false)

  *Constructor.*
- std::string **GetMsgType** () const

  *Get the typename of the message that is handled.*
- virtual bool **HandleData** (const std::string &_newdata, boost::function< void(uint32_t)> _cb, uint32_t _id)

  *Process new incoming data.*
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  *Process new incoming message.*
- virtual bool **IsLocal** () const

  *Is the callback local?*

### Additional Inherited Members

### 10.14.1 Detailed Description

**template<class M>class gazebo::transport::CallbackHelperT< M >**

Callback helper Template.

### 10.14.2 Constructor & Destructor Documentation

#### 10.14.2.1 template<class M > gazebo::transport::CallbackHelperT< M >::CallbackHelperT ( const boost::function< void(const boost::shared_ptr< M const > &)> & *_cb,* bool *_latching* = false ) [inline]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_cb* | boost function to call on incoming messages |
| in | *_latching* | Set to true to make the callback helper latching. |

### 10.14.3 Member Function Documentation

#### 10.14.3.1 template<class M > std::string gazebo::transport::CallbackHelperT< M >::GetMsgType ( ) const [inline], [virtual]

Get the typename of the message that is handled.

**Returns**

String representation of the message type

Reimplemented from **gazebo::transport::CallbackHelper** (p. 182).

References gzthrow, and NULL.

#### 10.14.3.2 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleData ( const std::string & *_newdata,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id* ) [inline], [virtual]

Process new incoming data.

**Parameters**

| | | |
|---|---|---|
| in | *_newdata* | Incoming data to be processed |

**Returns**

true if successfully processed; false otherwise

**Parameters**

| | | |
|---|---|---|
| in | *_cb* | If non-null, callback to be invoked which signals that transmission is complete. |
| in | *_id* | ID associated with the message data. |

Implements **gazebo::transport::CallbackHelper** (p. 182).

#### 10.14.3.3 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleMessage ( MessagePtr *_newMsg* ) [inline], [virtual]

Process new incoming message.

**Parameters**

| | | |
|---|---|---|
| in | *_newMsg* | Incoming message to be processed |

**Returns**

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 183).

#### 10.14.3.4 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::IsLocal ( ) const [inline], [virtual]

Is the callback local?

**Returns**

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 183).

The documentation for this class was generated from the following file:

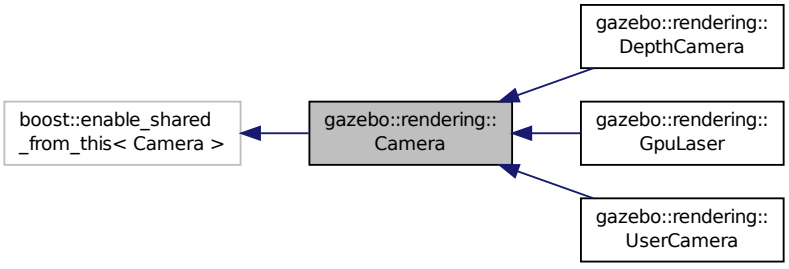
- **CallbackHelper.hh**

## 10.15 gazebo::rendering::Camera Class Reference

Basic camera sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Camera:




### Public Member Functions

- **Camera** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)

  *Constructor.*
- virtual ~**Camera** ()

*Destructor.*

- void **AttachToVisual** (const std::string &_visualName, bool _inheritOrientation, double _minDist=0.0, double _maxDist=0.0)

  *Attach the camera to a scene node.*
- void **AttachToVisual** (uint32_t _id, bool _inheritOrientation, double _minDist=0.0, double _maxDist=0.0)

  *Attach the camera to a scene node.*
- template<typename T >
  **event::ConnectionPtr ConnectNewImageFrame** (T _subscriber)

  *Connect to the new image signal.*
- void **CreateRenderTexture** (const std::string &_textureName)

  *Set the render target.*
- void **DisconnectNewImageFrame** (**event::ConnectionPtr** &_c)

  *Disconnect from an image frame.*
- void **EnableSaveFrame** (bool _enable)

  *Enable or disable saving.*
- virtual void **Fini** ()

  *Finalize the camera.*
- float **GetAspectRatio** () const

  *Get the apect ratio.*
- virtual float **GetAvgFPS** ()

  *Get the average FPS.*
- void **GetCameraToViewportRay** (int _screenx, int _screeny, **math::Vector3** &_origin, **math::Vector3** &_dir)

  *Get a world space ray as cast from the camera through the viewport.*
- bool **GetCaptureData** () const

  *Return the value of this->captureData.*
- **math::Vector3 GetDirection** () const

  *Get the camera's direction vector.*
- double **GetFarClip** ()

  *Get the far clip distance.*
- **math::Angle GetHFOV** () const

  *Get the camera FOV (horizontal)*
- size_t **GetImageByteSize** () const

  *Get the image size in bytes.*
- virtual const unsigned char * **GetImageData** (unsigned int i=0)

  *Get a pointer to the image data.*
- unsigned int **GetImageDepth** () const

  *Get the depth of the image.*
- std::string **GetImageFormat** () const

  *Get the string representation of the image format.*
- virtual unsigned int **GetImageHeight** () const

  *Get the height of the image.*
- virtual unsigned int **GetImageWidth** () const

  *Get the width of the image.*
- bool **GetInitialized** () const

  *Return true if the camera has been initialized.*
- **common::Time GetLastRenderWallTime** ()

  *Get the last time the camera was rendered.*

- std::string **GetName** () const

  *Get the camera's name.*
- double **GetNearClip** ()

  *Get the near clip distance.*
- Ogre::Camera * **GetOgreCamera** () const

  *Get a pointer to the ogre camera.*
- Ogre::SceneNode * **GetPitchNode** () const

  *Get the camera's pitch scene node.*
- double **GetRenderRate** () const

  *Get the render Hz rate.*
- Ogre::Texture * **GetRenderTexture** () const

  *Get the render texture.*
- **math::Vector3 GetRight** ()

  *Get the viewport right vector.*
- **ScenePtr GetScene** () const

  *Get the scene this camera is in.*
- Ogre::SceneNode * **GetSceneNode** () const

  *Get the camera's scene node.*
- std::string **GetScreenshotPath** () const

  *Get the path to saved screenshots.*
- unsigned int **GetTextureHeight** () const

  *Get the height of the off-screen render texture.*
- unsigned int **GetTextureWidth** () const

  *Get the width of the off-screen render texture.*
- virtual unsigned int **GetTriangleCount** ()

  *Get the triangle count.*
- **math::Vector3 GetUp** ()

  *Get the viewport up vector.*
- **math::Angle GetVFOV** () const

  *Get the camera FOV (vertical)*
- Ogre::Viewport * **GetViewport** () const

  *Get a pointer to the Ogre::Viewport.*
- unsigned int **GetViewportHeight** () const

  *Get the viewport height in pixels.*
- unsigned int **GetViewportWidth** () const

  *Get the viewport width in pixels.*
- unsigned int **GetWindowId** () const

  *Get the ID of the window this camera is rendering into.*
- bool **GetWorldPointOnPlane** (int _x, int _y, const **math::Plane** &_plane, **math::Vector3** &_result)

  *Get point on a plane.*
- **math::Pose GetWorldPose** ()

  *Get the global pose of the camera.*
- **math::Vector3 GetWorldPosition** () const

  *Get the camera position in the world.*
- **math::Quaternion GetWorldRotation** () const

  *Get the camera's orientation in the world.*
- double **GetZValue** (int _x, int _y)

*Get the Z-buffer value at the given image coordinate.*

- virtual void **Init** ()

  *Initialize the camera.*

- bool **IsAnimating** () const

  *Return true if the camera is moving due to an animation.*

- bool **IsVisible** (**VisualPtr** _visual)

  *Return true if the visual is within the camera's view frustum.*

- bool **IsVisible** (const std::string &_visualName)

  *Return true if the visual is within the camera's view frustum.*

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the camera with a set of parmeters.*

- virtual void **Load** ()

  *Load the camera with default parmeters.*

- virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)

  *Move the camera to a position (this is an animated motion).*

- bool **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _onComplete=**NULL**)

  *Move the camera to a series of poses (this is an animated motion).*

- virtual void **PostRender** ()

  *Post render.*

- void **Render** ()

  *Render the camera.*

- void **Render** (bool _force)

  *Render the camera.*

- void **RotatePitch** (**math::Angle** _angle)

  *Rotate the camera around the pitch axis.*

- void **RotateYaw** (**math::Angle** _angle)

  *Rotate the camera around the yaw axis.*

- bool **SaveFrame** (const std::string &_filename)

  *Save the last frame to disk.*

- void **SetAspectRatio** (float _ratio)

  *Set the aspect ratio.*

- void **SetCaptureData** (bool _value)

  *Set whether to capture data.*

- void **SetCaptureDataOnce** ()

  *Capture data once and save to disk.*

- void **SetClipDist** (float _near, float _far)

  *Set the clip distances.*

- void **SetHFOV** (**math::Angle** _angle)

  *Set the camera FOV (horizontal)*

- void **SetImageHeight** (unsigned int _h)

  *Set the image height.*

- void **SetImageSize** (unsigned int _w, unsigned int _h)

  *Set the image size.*

- void **SetImageWidth** (unsigned int _w)

  *Set the image height.*

- void **SetName** (const std::string &_name)

*Set the camera's name.*

- void **SetRenderRate** (double _hz)

  *Set the render Hz rate.*
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

  *Set the camera's render target.*
- void **SetSaveFramePathname** (const std::string &_pathname)

  *Set the save frame pathname.*
- void **SetScene** (**ScenePtr** _scene)

  *Set the scene this camera is viewing.*
- void **SetSceneNode** (Ogre::SceneNode *_node)

  *Set the camera's scene node.*
- void **SetWindowId** (unsigned int _windowId)
- virtual void **SetWorldPose** (const **math::Pose** &_pose)

  *Set the global pose of the camera.*
- void **SetWorldPosition** (const **math::Vector3** &_pos)

  *Set the world position.*
- void **SetWorldRotation** (const **math::Quaternion** &_quat)

  *Set the world orientation.*
- void **ShowWireframe** (bool _s)

  *Set whether to view the world in wireframe.*
- void **ToggleShowWireframe** ()

  *Toggle whether to view the world in wireframe.*
- void **TrackVisual** (const std::string &_visualName)

  *Set the camera to track a scene node.*
- void **Translate** (const **math::Vector3** &_direction)

  *Translate the camera.*
- virtual void **Update** ()

## Static Public Member Functions

- static size_t **GetImageByteSize** (unsigned int _width, unsigned int _height, const std::string &_format)

  *Calculate image byte size base on a few parameters.*
- static bool **SaveFrame** (const unsigned char *_image, unsigned int _width, unsigned int _height, int _depth, const std::string &_format, const std::string &_filename)

  *Save a frame using an image buffer.*

## Protected Member Functions

- virtual void **AnimationComplete** ()

  *Internal function used to indicate that an animation has completed.*
- virtual bool **AttachToVisualImpl** (const std::string &_name, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  *Attach the camera to a scene node.*
- virtual bool **AttachToVisualImpl** (uint32_t _id, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  *Attach the camera to a scene node.*
- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

*Attach the camera to a visual.*

- std::string **GetFrameFilename** ()

  *Get the next frame filename based on SDF parameters.*

- void **ReadPixelBuffer** ()

  *Read image data from pixel buffer.*

- virtual void **RenderImpl** ()

  *Implementation of the render call.*

- bool **TrackVisualImpl** (const std::string &_visualName)

  *Implementation of the **Camera::TrackVisual** (p. 209) call.*

- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

  *Set the camera to track a scene node.*

## Protected Attributes

- Ogre::AnimationState * **animState**

  *Animation state, used to animate the camera.*

- unsigned char * **bayerFrameBuffer**

  *Buffer for a bayer image frame.*

- Ogre::Camera * **camera**

  *The OGRE camera.*

- bool **captureData**

  *True to capture frames into an image buffer.*

- bool **captureDataOnce**

  *True to capture a frame once and save to disk.*

- std::vector< **event::ConnectionPtr** > **connections**

  *The camera's event connections.*

- int **imageFormat**

  *Format for saving images.*

- int **imageHeight**

  *Save image height.*

- int **imageWidth**

  *Save image width.*

- bool **initialized**

  *True if initialized.*

- **common::Time lastRenderWallTime**

  *Time the last frame was rendered.*

- std::string **name**

  *Name of the camera.*

- bool **newData**

  *True if new data is available.*

- **event::EventT**< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> **newImageFrame**

  *Event triggered when a new frame is generated.*

- boost::function< void()> **onAnimationComplete**

  *User callback for when an animation completes.*

- Ogre::SceneNode * **pitchNode**

  ***Scene*** *(p. 814) nod that controls camera pitch.*
- **common::Time prevAnimTime**

  *Previous time the camera animation was updated.*
- Ogre::RenderTarget * **renderTarget**

  *Target that renders frames.*
- Ogre::Texture * **renderTexture**

  *Texture that receives results from rendering.*
- std::list< msgs::Request > **requests**

  *List of requests.*
- unsigned int **saveCount**

  *Number of saved frames.*
- unsigned char * **saveFrameBuffer**
- **ScenePtr scene**

  *Pointer to the scene.*
- Ogre::SceneNode * **sceneNode**

  ***Scene*** *(p. 814) node that controls camera position.*
- std::string **screenshotPath**

  *Path to saved screenshots.*
- sdf::ElementPtr **sdf**

  ***Camera*** *(p. 186)'s SDF values.*
- unsigned int **textureHeight**

  *Height of the render texture.*
- unsigned int **textureWidth**

  *Width of the render texture.*
- Ogre::Viewport * **viewport**

  *Viewport the ogre camera uses.*
- unsigned int **windowId**

  *ID of the window that the camera is attached to.*

### 10.15.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

### 10.15.2 Constructor & Destructor Documentation

#### 10.15.2.1 gazebo::rendering::Camera::Camera ( const std::string & *_namePrefix,* ScenePtr *_scene,* bool *_autoRender* = `true` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_namePrefix* | Unique prefix name for the camera. |
| in | *_scene* | **Scene** (p. 814) that will contain the camera |
| in | *_autoRender* | Almost everyone should leave this as true. |

#### 10.15.2.2 virtual gazebo::rendering::Camera::~Camera ( ) [virtual]

Destructor.

### 10.15.3 Member Function Documentation

#### 10.15.3.1 virtual void gazebo::rendering::Camera::AnimationComplete ( ) [protected], [virtual]

Internal function used to indicate that an animation has completed.

Reimplemented in **gazebo::rendering::UserCamera** (p. 1068).

#### 10.15.3.2 void gazebo::rendering::Camera::AttachToVisual ( const std::string & *_visualName*, bool *_inheritOrientation*, double *_minDist* = 0.0, double *_maxDist* = 0.0 )

Attach the camera to a scene node.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to attach the camera to |
| in | *_inherit-Orientation* | True means camera acquires the visual's orientation |
| in | *_minDist* | Minimum distance the camera is allowed to get to the visual |
| in | *_maxDist* | Maximum distance the camera is allowd to get from the visual |

#### 10.15.3.3 void gazebo::rendering::Camera::AttachToVisual ( uint32_t *_id*, bool *_inheritOrientation*, double *_minDist* = 0.0, double *_maxDist* = 0.0 )

Attach the camera to a scene node.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the visual to attach the camera to |
| in | *_inherit-Orientation* | True means camera acquires the visual's orientation |
| in | *_minDist* | Minimum distance the camera is allowed to get to the visual |
| in | *_maxDist* | Maximum distance the camera is allowd to get from the visual |

#### 10.15.3.4 virtual bool gazebo::rendering::Camera::AttachToVisualImpl ( const std::string & *_name*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0 ) [protected], [virtual]

Attach the camera to a scene node.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to attach the camera to |
| in | *_inherit-Orientation* | True means camera acquires the visual's orientation |
| in | *_minDist* | Minimum distance the camera is allowed to get to the visual |
| in | *_maxDist* | Maximum distance the camera is allowd to get from the visual |

**Returns**

True on success

#### 10.15.3.5 virtual bool gazebo::rendering::Camera::AttachToVisualImpl ( uint32_t *_id*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0 ) [protected], [virtual]

Attach the camera to a scene node.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the visual to attach the camera to |
| in | *_inherit-Orientation* | True means camera acquires the visual's orientation |
| in | *_minDist* | Minimum distance the camera is allowed to get to the visual |
| in | *_maxDist* | Maximum distance the camera is allowd to get from the visual |

**Returns**

True on success

#### 10.15.3.6 virtual bool gazebo::rendering::Camera::AttachToVisualImpl ( VisualPtr *_visual*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0 ) [protected], [virtual]

Attach the camera to a visual.

**Parameters**

| | | |
|---|---|---|
| in | *_visual* | The visual to attach the camera to |
| in | *_inherit-Orientation* | True means camera acquires the visual's orientation |
| in | *_minDist* | Minimum distance the camera is allowed to get to the visual |
| in | *_maxDist* | Maximum distance the camera is allowd to get from the visual |

**Returns**

True on success

Reimplemented in **gazebo::rendering::UserCamera** (p. 1069).

#### 10.15.3.7 template<typename T > event::ConnectionPtr gazebo::rendering::Camera::ConnectNewImageFrame ( T *_subscriber* ) [inline]

Connect to the new image signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback that is called when a new image is generated |

**Returns**

A pointer to the connection. This must be kept in scope.

References gazebo::event::EventT< T >::Connect(), and newImageFrame.

#### 10.15.3.8 void gazebo::rendering::Camera::CreateRenderTexture ( const std::string & *_textureName* )

Set the render target.

**Parameters**

| | | |
|---|---|---|
| in | *_textureName* | Name of the new render texture |

#### 10.15.3.9 void gazebo::rendering::Camera::DisconnectNewImageFrame ( event::ConnectionPtr & *_c* ) [inline]

Disconnect from an image frame.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The connection to disconnect |

References gazebo::event::EventT< T >::Disconnect(), and newImageFrame.

#### 10.15.3.10 void gazebo::rendering::Camera::EnableSaveFrame ( bool *_enable* )

Enable or disable saving.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | Set to True to enable saving of frames |

#### 10.15.3.11 virtual void gazebo::rendering::Camera::Fini ( ) [virtual]

Finalize the camera.

This function is called before the camera is destructed

Reimplemented in **gazebo::rendering::GpuLaser** (p. 433), **gazebo::rendering::DepthCamera** (p. 361), and **gazebo::rendering::UserCamera** (p. 1069).

#### 10.15.3.12 float gazebo::rendering::Camera::GetAspectRatio ( ) const

Get the apect ratio.

**Returns**

The aspect ratio (width / height) in pixels

#### 10.15.3.13 virtual float gazebo::rendering::Camera::GetAvgFPS ( ) [inline], [virtual]

Get the average FPS.

**Returns**

The average frames per second

#### 10.15.3.14 void gazebo::rendering::Camera::GetCameraToViewportRay ( int *_screenx*, int *_screeny*, math::Vector3 & *_origin*, math::Vector3 & *_dir* )

Get a world space ray as cast from the camera through the viewport.

**Parameters**

| | | |
|---|---|---|
| in | *_screenx* | X coordinate in the camera's viewport, in pixels. |
| in | *_screeny* | Y coordinate in the camera's viewport, in pixels. |
| out | *_origin* | Origin in the world coordinate frame of the resulting ray |
| out | *_dir* | Direction of the resulting ray |

#### 10.15.3.15 bool gazebo::rendering::Camera::GetCaptureData ( ) const

Return the value of this->captureData.

**Returns**

True if the camera is set to capture data.

#### 10.15.3.16 math::Vector3 gazebo::rendering::Camera::GetDirection ( ) const

Get the camera's direction vector.

**Returns**

Direction the camera is facing

#### 10.15.3.17 double gazebo::rendering::Camera::GetFarClip ( )

Get the far clip distance.

**Returns**

Far clip distance

#### 10.15.3.18 std::string gazebo::rendering::Camera::GetFrameFilename ( ) [protected]

Get the next frame filename based on SDF parameters.

**Returns**

The frame's filename

#### 10.15.3.19 math::Angle gazebo::rendering::Camera::GetHFOV ( ) const

Get the camera FOV (horizontal)

**Returns**

The horizontal field of view

#### 10.15.3.20 size_t gazebo::rendering::Camera::GetImageByteSize ( ) const

Get the image size in bytes.

**Returns**

Size in bytes

#### 10.15.3.21 static size_t gazebo::rendering::Camera::GetImageByteSize ( unsigned int *_width*, unsigned int *_height*, const std::string & *_format* ) [static]

Calculate image byte size base on a few parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_width* | Width of an image |
| in | *_height* | Height of an image |
| in | *_format* | Image format |

**Returns**

Size of an image based on the parameters

#### 10.15.3.22 virtual const unsigned char* gazebo::rendering::Camera::GetImageData ( unsigned int *i* = 0 ) [virtual]

Get a pointer to the image data.

Get the raw image data from a camera's buffer.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | Index of the camera's texture (0 = RGB, 1 = depth). |

**Returns**

Pointer to the raw data, null if data is not available.

#### 10.15.3.23 unsigned int gazebo::rendering::Camera::GetImageDepth ( ) const

Get the depth of the image.

**Returns**

Depth of the image

#### 10.15.3.24 std::string gazebo::rendering::Camera::GetImageFormat ( ) const

Get the string representation of the image format.

**Returns**

String representation of the image format.

#### 10.15.3.25 virtual unsigned int gazebo::rendering::Camera::GetImageHeight ( ) const [virtual]

Get the height of the image.

**Returns**

Image height

Reimplemented in **gazebo::rendering::UserCamera** (p. 1070).

#### 10.15.3.26 virtual unsigned int gazebo::rendering::Camera::GetImageWidth ( ) const [virtual]

Get the width of the image.

**Returns**

Image width

Reimplemented in **gazebo::rendering::UserCamera** (p. 1070).

#### 10.15.3.27 bool gazebo::rendering::Camera::GetInitialized ( ) const

Return true if the camera has been initialized.

**Returns**

True if initialized was successful

#### 10.15.3.28 common::Time gazebo::rendering::Camera::GetLastRenderWallTime ( )

Get the last time the camera was rendered.

**Returns**

Time the camera was last rendered

#### 10.15.3.29 std::string gazebo::rendering::Camera::GetName ( ) const

Get the camera's name.

**Returns**

The name of the camera

#### 10.15.3.30 double gazebo::rendering::Camera::GetNearClip ( )

Get the near clip distance.

**Returns**

Near clip distance

#### 10.15.3.31 Ogre::Camera* gazebo::rendering::Camera::GetOgreCamera ( ) const

Get a pointer to the ogre camera.

**Returns**

Pointer to the OGRE camera

#### 10.15.3.32 Ogre::SceneNode* gazebo::rendering::Camera::GetPitchNode ( ) const

Get the camera's pitch scene node.

**Returns**

The pitch node the camera is attached to

#### 10.15.3.33 double gazebo::rendering::Camera::GetRenderRate ( ) const

Get the render Hz rate.

**Returns**

The Hz rate

#### 10.15.3.34 Ogre::Texture* gazebo::rendering::Camera::GetRenderTexture ( ) const

Get the render texture.

**Returns**

Pointer to the render texture

#### 10.15.3.35 math::Vector3 gazebo::rendering::Camera::GetRight ( )

Get the viewport right vector.

**Returns**

The viewport right vector

#### 10.15.3.36 ScenePtr gazebo::rendering::Camera::GetScene ( ) const

Get the scene this camera is in.

**Returns**

Pointer to scene containing this camera

#### 10.15.3.37 Ogre::SceneNode* gazebo::rendering::Camera::GetSceneNode ( ) const

Get the camera's scene node.

**Returns**

The scene node the camera is attached to

#### 10.15.3.38 std::string gazebo::rendering::Camera::GetScreenshotPath ( ) const

Get the path to saved screenshots.

**Returns**

Path to saved screenshots.

#### 10.15.3.39 unsigned int gazebo::rendering::Camera::GetTextureHeight ( ) const

Get the height of the off-screen render texture.

**Returns**

Render texture height

#### 10.15.3.40 unsigned int gazebo::rendering::Camera::GetTextureWidth ( ) const

Get the width of the off-screen render texture.

**Returns**

Render texture width

#### 10.15.3.41 virtual unsigned int gazebo::rendering::Camera::GetTriangleCount ( ) [inline], [virtual]

Get the triangle count.

**Returns**

The current triangle count

#### 10.15.3.42 math::Vector3 gazebo::rendering::Camera::GetUp ( )

Get the viewport up vector.

**Returns**

The viewport up vector

#### 10.15.3.43 math::Angle gazebo::rendering::Camera::GetVFOV ( ) const

Get the camera FOV (vertical)

**Returns**

The vertical field of view

#### 10.15.3.44 Ogre::Viewport* gazebo::rendering::Camera::GetViewport ( ) const

Get a pointer to the Ogre::Viewport.

**Returns**

Pointer to the Ogre::Viewport

#### 10.15.3.45 unsigned int gazebo::rendering::Camera::GetViewportHeight ( ) const

Get the viewport height in pixels.

**Returns**

The viewport height

#### 10.15.3.46 unsigned int gazebo::rendering::Camera::GetViewportWidth ( ) const

Get the viewport width in pixels.

**Returns**

The viewport width

#### 10.15.3.47 unsigned int gazebo::rendering::Camera::GetWindowId ( ) const

Get the ID of the window this camera is rendering into.

**Returns**

The ID of the window.

#### 10.15.3.48 bool gazebo::rendering::Camera::GetWorldPointOnPlane ( int *_x*, int *_y*, const **math::Plane** & *_plane*, **math::Vector3** & *_result* )

Get point on a plane.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | X cooridnate in camera's viewport, in pixels |
| in | *_y* | Y cooridnate in camera's viewport, in pixels |
| in | *_plane* | Plane on which to find the intersecting point |
| out | *_result* | Point on the plane |

**Returns**

True if a valid point was found

#### 10.15.3.49 math::Pose gazebo::rendering::Camera::GetWorldPose ( )

Get the global pose of the camera.

**Returns**

Pose of the camera in the world coordinate frame

#### 10.15.3.50 math::Vector3 gazebo::rendering::Camera::GetWorldPosition ( ) const

Get the camera position in the world.

**Returns**

The world position of the camera

#### 10.15.3.51 math::Quaternion gazebo::rendering::Camera::GetWorldRotation ( ) const

Get the camera's orientation in the world.

**Returns**

The camera's orientation as a **math::Quaternion** (p. 761)

#### 10.15.3.52 double gazebo::rendering::Camera::GetZValue ( int _x, int _y )

Get the Z-buffer value at the given image coordinate.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | Image coordinate; (0, 0) specifies the top-left corner. |
| in | *_y* | Image coordinate; (0, 0) specifies the top-left corner. |

**Returns**

Image z value; note that this is abitrarily scaled and is *not* the same as the depth value.

#### 10.15.3.53 virtual void gazebo::rendering::Camera::Init ( ) [virtual]

Initialize the camera.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 435), **gazebo::rendering::DepthCamera** (p. 361), and **gazebo::rendering::UserCamera** (p. 1071).

#### 10.15.3.54 bool gazebo::rendering::Camera::IsAnimating ( ) const

Return true if the camera is moving due to an animation.

#### 10.15.3.55 bool gazebo::rendering::Camera::IsVisible ( VisualPtr _visual )

Return true if the visual is within the camera's view frustum.

**Parameters**

| | | |
|---|---|---|
| in | *_visual* | The visual to check for visibility |

**Returns**

True if the _visual is in the camera's frustum

#### 10.15.3.56 bool gazebo::rendering::Camera::IsVisible ( const std::string & _visualName )

Return true if the visual is within the camera's view frustum.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to check for visibility |

**Returns**

True if the _visual is in the camera's frustum

#### 10.15.3.57 virtual void gazebo::rendering::Camera::Load ( sdf::ElementPtr _sdf ) [virtual]

Load the camera with a set of parmeters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The SDF camera info |

Reimplemented in **gazebo::rendering::UserCamera** (p. 1071).

#### 10.15.3.58 virtual void gazebo::rendering::Camera::Load ( ) [virtual]

Load the camera with default parmeters.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 435), **gazebo::rendering::DepthCamera** (p. 361), and **gazebo::rendering::UserCamera** (p. 1072).

#### 10.15.3.59 virtual bool gazebo::rendering::Camera::MoveToPosition ( const math::Pose & _pose, double _time ) [virtual]

Move the camera to a position (this is an animated motion).

**See Also**

**Camera::MoveToPositions** (p. 204)

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | End position of the camera |
| in | *_time* | Duration of the camera's movement |

Reimplemented in **gazebo::rendering::UserCamera** (p. 1072).

#### 10.15.3.60 bool gazebo::rendering::Camera::MoveToPositions ( const std::vector< math::Pose > & _pts, double _time, boost::function< void() > _onComplete = NULL )

Move the camera to a series of poses (this is an animated motion).

**See Also**

**Camera::MoveToPosition** (p. 204)

**Parameters**

| | | |
|---|---|---|
| in | *_pts* | Vector of poses to move to |
| in | *_time* | Duration of the entire move |
| in | *_onComplete* | Callback that is called when the move is complete |

#### 10.15.3.61 virtual void gazebo::rendering::Camera::PostRender ( ) [virtual]

Post render.

Called afer the render signal.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 435), **gazebo::rendering::DepthCamera** (p. 361), and **gazebo::rendering::UserCamera** (p. 1072).

#### 10.15.3.62 void gazebo::rendering::Camera::ReadPixelBuffer ( ) [protected]

Read image data from pixel buffer.

#### 10.15.3.63 void gazebo::rendering::Camera::Render ( )

Render the camera.

Called after the pre-render signal. This function will generate camera images.

#### 10.15.3.64 void gazebo::rendering::Camera::Render ( bool *_force* )

Render the camera.

Called after the pre-render signal. This function will generate camera images.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force camera to render. Ignore camera update rate. |

#### 10.15.3.65 virtual void gazebo::rendering::Camera::RenderImpl ( ) [protected], [virtual]

Implementation of the render call.

#### 10.15.3.66 void gazebo::rendering::Camera::RotatePitch ( math::Angle *_angle* )

Rotate the camera around the pitch axis.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | Pitch amount |

#### 10.15.3.67 void gazebo::rendering::Camera::RotateYaw ( math::Angle *_angle* )

Rotate the camera around the yaw axis.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | Rotation amount |

#### 10.15.3.68 bool gazebo::rendering::Camera::SaveFrame ( const std::string & *_filename* )

Save the last frame to disk.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | File in which to save a single frame |

**Returns**

True if saving was successful

#### 10.15.3.69 static bool gazebo::rendering::Camera::SaveFrame ( const unsigned char * *_image*, unsigned int *_width*, unsigned int *_height*, int *_depth*, const std::string & *_format*, const std::string & *_filename* ) [static]

Save a frame using an image buffer.

**Parameters**

| | | |
|---|---|---|
| in | *_image* | The raw image buffer |
| in | *_width* | Width of the image |
| in | *_height* | Height of the image |
| in | *_depth* | Depth of the image data |
| in | *_format* | Format the image data is in |
| in | *_filename* | Name of the file in which to write the frame |

**Returns**

True if saving was successful

#### 10.15.3.70 void gazebo::rendering::Camera::SetAspectRatio ( float *_ratio* )

Set the aspect ratio.

**Parameters**

| | | |
|---|---|---|
| in | *_ratio* | The aspect ratio (width / height) in pixels |

#### 10.15.3.71 void gazebo::rendering::Camera::SetCaptureData ( bool *_value* )

Set whether to capture data.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | Set to true to capture data into a memory buffer. |

#### 10.15.3.72 void gazebo::rendering::Camera::SetCaptureDataOnce ( )

Capture data once and save to disk.

#### 10.15.3.73 void gazebo::rendering::Camera::SetClipDist ( float *_near*, float *_far* )

Set the clip distances.

**Parameters**

| | | |
|---|---|---|
| in | *_near* | Near clip distance in meters |
| in | *_far* | Far clip distance in meters |

#### 10.15.3.74 void gazebo::rendering::Camera::SetHFOV ( math::Angle *_angle* )

Set the camera FOV (horizontal)

**Parameters**

| | | |
|---|---|---|
| in | *_radians* | Horizontal field of view |

#### 10.15.3.75 void gazebo::rendering::Camera::SetImageHeight ( unsigned int *_h* )

Set the image height.

**Parameters**

| | | |
|---|---|---|
| in | *_h* | Image height |

#### 10.15.3.76 void gazebo::rendering::Camera::SetImageSize ( unsigned int *_w*, unsigned int *_h* )

Set the image size.

**Parameters**

| | | |
|---|---|---|
| in | *_w* | Image width |
| in | *_h* | Image height |

#### 10.15.3.77 void gazebo::rendering::Camera::SetImageWidth ( unsigned int *_w* )

Set the image height.

**Parameters**

| | | |
|---|---|---|
| in | *_w* | Image width |

#### 10.15.3.78 void gazebo::rendering::Camera::SetName ( const std::string & *_name* )

Set the camera's name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | New name for the camera |

#### 10.15.3.79 void gazebo::rendering::Camera::SetRenderRate ( double *_hz* )

Set the render Hz rate.

**Parameters**

| | | |
|---|---|---|
| in | *_hz* | The Hz rate |

#### 10.15.3.80 virtual void gazebo::rendering::Camera::SetRenderTarget ( Ogre::RenderTarget * *_target* ) [virtual]

Set the camera's render target.

**Parameters**

| | | |
|---|---|---|
| in | *_target* | Pointer to the render target |

Reimplemented in **gazebo::rendering::UserCamera** (p. 1073).

#### 10.15.3.81 void gazebo::rendering::Camera::SetSaveFramePathname ( const std::string & *_pathname* )

Set the save frame pathname.

**Parameters**

| | | |
|---|---|---|
| in | *_pathname* | Directory in which to store saved image frames |

#### 10.15.3.82 void gazebo::rendering::Camera::SetScene ( ScenePtr *_scene* )

Set the scene this camera is viewing.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | Pointer to the scene |

#### 10.15.3.83 void gazebo::rendering::Camera::SetSceneNode ( Ogre::SceneNode * *_node* )

Set the camera's scene node.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | The scene nodes to attach the camera to |

#### 10.15.3.84 void gazebo::rendering::Camera::SetWindowId ( unsigned int *_windowId* )

#### 10.15.3.85 virtual void gazebo::rendering::Camera::SetWorldPose ( const math::Pose & *_pose* ) [virtual]

Set the global pose of the camera.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | The new **math::Pose** (p. 734) of the camera |

Reimplemented in **gazebo::rendering::UserCamera** (p. 1074).

#### 10.15.3.86 void gazebo::rendering::Camera::SetWorldPosition ( const math::Vector3 & *_pos* )

Set the world position.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | The new position of the camera |

#### 10.15.3.87 void gazebo::rendering::Camera::SetWorldRotation ( const math::Quaternion & *_quat* )

Set the world orientation.

**Parameters**

| | | |
|---|---|---|
| in | *_quat* | The new orientation of the camera |

#### 10.15.3.88 void gazebo::rendering::Camera::ShowWireframe ( bool *_s* )

Set whether to view the world in wireframe.

**Parameters**

| | | |
|---|---|---|
| in | *_s* | Set to True to render objects as wireframe |

#### 10.15.3.89 void gazebo::rendering::Camera::ToggleShowWireframe ( )

Toggle whether to view the world in wireframe.

#### 10.15.3.90 void gazebo::rendering::Camera::TrackVisual ( const std::string & *_visualName* )

Set the camera to track a scene node.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to track |

#### 10.15.3.91 bool gazebo::rendering::Camera::TrackVisualImpl ( const std::string & *_visualName* ) [protected]

Implementation of the **Camera::TrackVisual** (p. 209) call.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to track |

**Returns**

True if able to track the visual

#### 10.15.3.92 virtual bool gazebo::rendering::Camera::TrackVisualImpl ( VisualPtr *_visual* ) [protected], [virtual]

Set the camera to track a scene node.

**Parameters**

| | | |
|---|---|---|
| in | *_visual* | The visual to track |

**Returns**

True if able to track the visual

Reimplemented in **gazebo::rendering::UserCamera** (p. 1074).

#### 10.15.3.93 void gazebo::rendering::Camera::Translate ( const math::Vector3 & *_direction* )

Translate the camera.

**Parameters**

| | | |
|---|---|---|
| in | *_direction* | The translation vector |

#### 10.15.3.94 virtual void gazebo::rendering::Camera::Update ( ) [virtual]

Reimplemented in **gazebo::rendering::UserCamera** (p. 1074).

### 10.15.4 Member Data Documentation

#### 10.15.4.1 Ogre::AnimationState* gazebo::rendering::Camera::animState [protected]

Animation state, used to animate the camera.

#### 10.15.4.2 unsigned char* gazebo::rendering::Camera::bayerFrameBuffer [protected]

Buffer for a bayer image frame.

#### 10.15.4.3 Ogre::Camera* gazebo::rendering::Camera::camera [protected]

The OGRE camera.

#### 10.15.4.4 bool gazebo::rendering::Camera::captureData [protected]

True to capture frames into an image buffer.

#### 10.15.4.5 bool gazebo::rendering::Camera::captureDataOnce [protected]

True to capture a frame once and save to disk.

#### 10.15.4.6 std::vector<event::ConnectionPtr> gazebo::rendering::Camera::connections [protected]

The camera's event connections.

#### 10.15.4.7 int gazebo::rendering::Camera::imageFormat [protected]

Format for saving images.

#### 10.15.4.8 int gazebo::rendering::Camera::imageHeight [protected]

Save image height.

#### 10.15.4.9 int gazebo::rendering::Camera::imageWidth [protected]

Save image width.

#### 10.15.4.10 bool gazebo::rendering::Camera::initialized [protected]

True if initialized.

#### 10.15.4.11 common::Time gazebo::rendering::Camera::lastRenderWallTime [protected]

Time the last frame was rendered.

#### 10.15.4.12 std::string gazebo::rendering::Camera::name [protected]

Name of the camera.

#### 10.15.4.13 bool gazebo::rendering::Camera::newData [protected]

True if new data is available.

#### 10.15.4.14 event::EventT<void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> gazebo::rendering::Camera::newImageFrame [protected]

Event triggered when a new frame is generated.

Referenced by ConnectNewImageFrame(), and DisconnectNewImageFrame().

#### 10.15.4.15 boost::function<void()> gazebo::rendering::Camera::onAnimationComplete [protected]

User callback for when an animation completes.

#### 10.15.4.16 Ogre::SceneNode* gazebo::rendering::Camera::pitchNode [protected]

**Scene** (p. 814) nod that controls camera pitch.

#### 10.15.4.17 common::Time gazebo::rendering::Camera::prevAnimTime [protected]

Previous time the camera animation was updated.

#### 10.15.4.18 Ogre::RenderTarget* gazebo::rendering::Camera::renderTarget [protected]

Target that renders frames.

#### 10.15.4.19 Ogre::Texture* gazebo::rendering::Camera::renderTexture [protected]

Texture that receives results from rendering.

#### 10.15.4.20 std::list<msgs::Request> gazebo::rendering::Camera::requests [protected]

List of requests.

#### 10.15.4.21 unsigned int gazebo::rendering::Camera::saveCount [protected]

Number of saved frames.

#### 10.15.4.22 unsigned char* gazebo::rendering::Camera::saveFrameBuffer [protected]

#### 10.15.4.23 ScenePtr gazebo::rendering::Camera::scene [protected]

Pointer to the scene.

#### 10.15.4.24 Ogre::SceneNode* gazebo::rendering::Camera::sceneNode [protected]

**Scene** (p. 814) node that controls camera position.

#### 10.15.4.25 std::string gazebo::rendering::Camera::screenshotPath [protected]

Path to saved screenshots.

#### 10.15.4.26 sdf::ElementPtr gazebo::rendering::Camera::sdf [protected]

**Camera** (p. 186)'s SDF values.

#### 10.15.4.27 unsigned int gazebo::rendering::Camera::textureHeight [protected]

Height of the render texture.

#### 10.15.4.28 unsigned int gazebo::rendering::Camera::textureWidth [protected]

Width of the render texture.

#### 10.15.4.29 Ogre::Viewport* gazebo::rendering::Camera::viewport [protected]

Viewport the ogre camera uses.

#### 10.15.4.30 unsigned int gazebo::rendering::Camera::windowId [protected]

ID of the window that the camera is attached to.

The documentation for this class was generated from the following file:

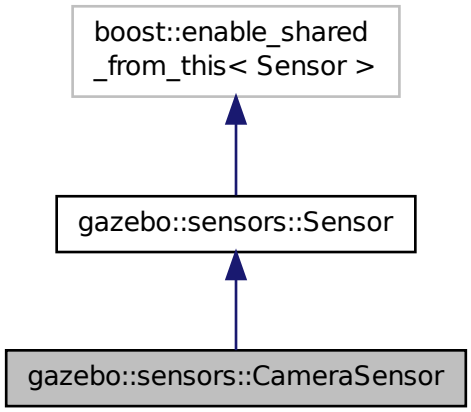
- **Camera.hh**

## 10.16 gazebo::sensors::CameraSensor Class Reference

Basic camera sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::CameraSensor:




### Public Member Functions

- **CameraSensor** ()

  *Constructor.*
- virtual ~**CameraSensor** ()

  *Destructor.*

- **rendering::CameraPtr GetCamera** () const

  *Returns a pointer to the **rendering::Camera** (p. 186).*
- const unsigned char * **GetImageData** ()

  *Gets the raw image data from the sensor.*
- unsigned int **GetImageHeight** () const

  *Gets the height of the image in pixels.*
- unsigned int **GetImageWidth** () const

  *Gets the width of the image in pixels.*
- virtual std::string **GetTopic** () const

  *Gets the topic name of the sensor.*
- virtual void **Init** ()

  *Initialize the camera.*
- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  *Load the sensor with SDF parameters.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*
- bool **SaveFrame** (const std::string &_filename)

  *Saves the image to the disk.*

### Protected Member Functions

- virtual void **Fini** ()

  *Finalize the camera.*
- virtual void **UpdateImpl** (bool _force)

  *Update the sensor information.*

### Additional Inherited Members

### 10.16.1 Detailed Description

Basic camera sensor.

This sensor is used for simulating standard monocular cameras

### 10.16.2 Constructor & Destructor Documentation

#### 10.16.2.1 gazebo::sensors::CameraSensor::CameraSensor ( )

Constructor.

#### 10.16.2.2 virtual gazebo::sensors::CameraSensor::~CameraSensor ( ) [virtual]

Destructor.

### 10.16.3 Member Function Documentation

#### 10.16.3.1 virtual void gazebo::sensors::CameraSensor::Fini ( ) [protected], [virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.16.3.2 rendering::CameraPtr gazebo::sensors::CameraSensor::GetCamera ( ) const [inline]

Returns a pointer to the **rendering::Camera** (p. 186).

**Returns**

The Pointer to the camera sensor.

#### 10.16.3.3 const unsigned char* gazebo::sensors::CameraSensor::GetImageData ( )

Gets the raw image data from the sensor.

**Returns**

The pointer to the image data array.

#### 10.16.3.4 unsigned int gazebo::sensors::CameraSensor::GetImageHeight ( ) const

Gets the height of the image in pixels.

**Returns**

The image height in pixels.

#### 10.16.3.5 unsigned int gazebo::sensors::CameraSensor::GetImageWidth ( ) const

Gets the width of the image in pixels.

**Returns**

The image width in pixels.

#### 10.16.3.6 virtual std::string gazebo::sensors::CameraSensor::GetTopic ( ) const [virtual]

Gets the topic name of the sensor.

**Returns**

Topic name

**Todo** to be implemented

Reimplemented from **gazebo::sensors::Sensor** (p. 843).

#### 10.16.3.7 virtual void gazebo::sensors::CameraSensor::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.16.3.8 virtual bool gazebo::sensors::CameraSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.16.3.9 virtual void gazebo::sensors::CameraSensor::Load ( const std::string & *_worldName,* sdf::ElementPtr *_sdf* ) [virtual]

Load the sensor with SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF **Sensor** (p. 837) parameters |
| in | *_worldName* | Name of world to load from |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.16.3.10 virtual void gazebo::sensors::CameraSensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.16.3.11 bool gazebo::sensors::CameraSensor::SaveFrame ( const std::string & *_filename* )

Saves the image to the disk.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | The name of the file to be saved. |

**Returns**

True if successful, false if unsuccessful.

#### 10.16.3.12 virtual void gazebo::sensors::CameraSensor::UpdateImpl ( bool *_force* ) [protected], [virtual]

Update the sensor information.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

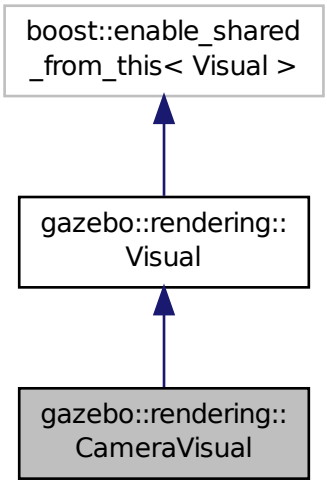
- **CameraSensor.hh**

## 10.17 gazebo::rendering::CameraVisual Class Reference

Basic camera visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::CameraVisual:




### Public Member Functions

- **CameraVisual** (const std::string &_name, **VisualPtr** _vis)
  *Constructor.*
- virtual ~**CameraVisual** ()
  *Destructor.*
- void **Load** (unsigned int _width, unsigned int _height)
  *Load the **Visual** (p. 1121).*

### Additional Inherited Members

### 10.17.1 Detailed Description

Basic camera visualization.

This class is used to visualize a camera image generated from a CameraSensor. The sensor's image is drawn on a billboard in the 3D environment.

### 10.17.2 Constructor & Destructor Documentation

#### 10.17.2.1 gazebo::rendering::CameraVisual::CameraVisual ( const std::string & *_name,* VisualPtr *_vis* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the **Visual** (p. 1121) |
| in | *_vis* | Pointer to the parent **Visual** (p. 1121) |

#### 10.17.2.2 virtual gazebo::rendering::CameraVisual::~CameraVisual ( ) [virtual]

Destructor.

### 10.17.3 Member Function Documentation

#### 10.17.3.1 void gazebo::rendering::CameraVisual::Load ( unsigned int *_width,* unsigned int *_height* )

Load the **Visual** (p. 1121).

**Parameters**

| | | |
|---|---|---|
| in | *_width* | Width of the **Camera** (p. 186) image |
| in | *_height* | Height of the **Camera** (p. 186) image |

The documentation for this class was generated from the following file:

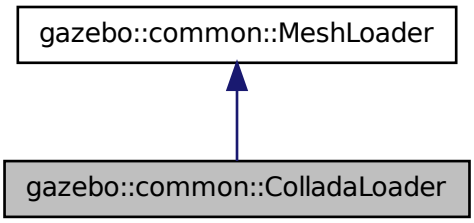
- **CameraVisual.hh**

## 10.18 gazebo::common::ColladaLoader Class Reference

Class used to load Collada mesh files.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::ColladaLoader:




### Public Member Functions

- **ColladaLoader** ()

  *Constructor.*
- virtual ~**ColladaLoader** ()

  *Destructor.*
- virtual **Mesh** * **Load** (const std::string &_filename)

  *Load a mesh.*

## 10.18.1 Detailed Description

Class used to load Collada mesh files.

## 10.18.2 Constructor & Destructor Documentation

### 10.18.2.1 gazebo::common::ColladaLoader::ColladaLoader ( )

Constructor.

### 10.18.2.2 virtual gazebo::common::ColladaLoader::~ColladaLoader ( ) [virtual]

Destructor.

## 10.18.3 Member Function Documentation

### 10.18.3.1 virtual Mesh* gazebo::common::ColladaLoader::Load ( const std::string & *_filename* ) [virtual]

Load a mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | Collada file to load |

**Returns**

Pointer to a new **Mesh** (p. 606)

Implements **gazebo::common::MeshLoader** (p. 615).

The documentation for this class was generated from the following file:

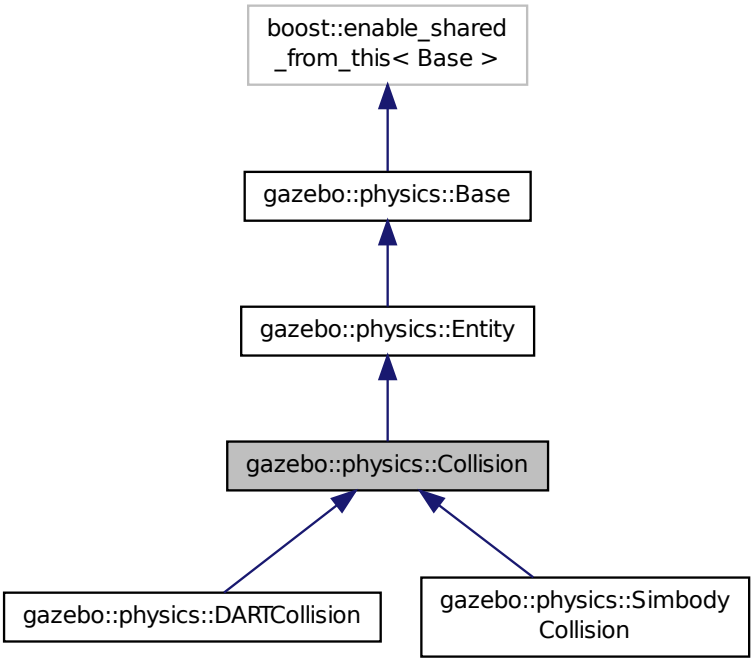
- **ColladaLoader.hh**

## 10.19 gazebo::physics::Collision Class Reference

**Base** (p. 159) class for all collision entities.

`#include <Collision.hh>`

Inheritance diagram for gazebo::physics::Collision:




### Public Member Functions

- **Collision** (**LinkPtr** _link)

  *Constructor.*
- virtual ~**Collision** ()

  *Destructor.*
- void **AddContact** (const **Contact** &_contact) **GAZEBO_DEPRECATED**(2.0)

*Add an occurance of a contact to this collision.*

- void **FillMsg** (msgs::Collision &_msg)

  *Fill a collision message.*
- virtual void **Fini** ()

  *Finalize the collision.*
- virtual **math::Box GetBoundingBox** () const =0

  *Get the bounding box for this collision.*
- bool **GetContactsEnabled** () const **GAZEBO_DEPRECATED**(2.0)

  *Return true of contacts are on.*
- float **GetLaserRetro** () const

  *Get the laser retro reflectiveness.*
- **LinkPtr GetLink** () const

  *Get the link this collision belongs to.*
- virtual int **GetMaxContacts** ()

  *returns number of contacts allowed for this collision.*
- **ModelPtr GetModel** () const

  *Get the model this collision belongs to.*
- virtual **math::Vector3 GetRelativeAngularAccel** () const

  *Get the angular acceleration of the collision.*
- virtual **math::Vector3 GetRelativeAngularVel** () const

  *Get the angular velocity of the collision.*
- virtual **math::Vector3 GetRelativeLinearAccel** () const

  *Get the linear acceleration of the collision.*
- virtual **math::Vector3 GetRelativeLinearVel** () const

  *Get the linear velocity of the collision.*
- **ShapePtr GetShape** () const

  *Get the collision shape.*
- unsigned int **GetShapeType** ()

  *Get the shape type.*
- **CollisionState GetState** ()

  *Get the collision state.*
- **SurfaceParamsPtr GetSurface** () const

  *Get the surface parameters.*
- virtual **math::Vector3 GetWorldAngularAccel** () const

  *Get the angular acceleration of the collision in the world frame.*
- virtual **math::Vector3 GetWorldAngularVel** () const

  *Get the angular velocity of the collision in the world frame.*
- virtual **math::Vector3 GetWorldLinearAccel** () const

  *Get the linear acceleration of the collision in the world frame.*
- virtual **math::Vector3 GetWorldLinearVel** () const

  *Get the linear velocity of the collision in the world frame.*
- virtual void **Init** ()

  *Initialize the collision.*
- bool **IsPlaceable** () const

  *Return whether this collision is movable.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the collision.*

- void **ProcessMsg** (const msgs::Collision &_msg)

  *Update parameters from a message.*
- virtual void **SetCategoryBits** (unsigned int _bits)=0

  *Set the category bits, used during collision detection.*
- virtual void **SetCollideBits** (unsigned int _bits)=0

  *Set the collide bits, used during collision detection.*
- void **SetCollision** (bool _placeable)

  *Set the encapsulated collsion object.*
- void **SetContactsEnabled** (bool _enable) **GAZEBO_DEPRECATED**(2.0)

  *Turn contact recording on or off.*
- void **SetLaserRetro** (float _retro)

  *Set the laser retro reflectiveness.*
- virtual void **SetMaxContacts** (double _maxContacts)

  *Number of contacts allowed for this collision.*
- void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the collision.*
- void **SetShape** (**ShapePtr** _shape)

  *Set the shape for this collision.*
- void **SetState** (const **CollisionState** &_state)

  *Set the current collision state.*
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  *Update the parameters using new sdf values.*

### Protected Attributes

- **LinkPtr link**

  *The link this collision belongs to.*
- bool **placeable**

  *Flag for placeable.*
- **ShapePtr shape**

  *Pointer to **physics::Shape** (p. 861).*

### Additional Inherited Members

### 10.19.1 Detailed Description

**Base** (p. 159) class for all collision entities.

### 10.19.2 Constructor & Destructor Documentation

#### 10.19.2.1 gazebo::physics::Collision::Collision ( **LinkPtr** *_link* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_link* | **Link** (p. 542) that contains this collision object. |

#### 10.19.2.2 virtual gazebo::physics::Collision::~Collision ( ) [virtual]

Destructor.

### 10.19.3 Member Function Documentation

#### 10.19.3.1 void gazebo::physics::Collision::AddContact ( const Contact & _contact )

Add an occurance of a contact to this collision.

Deprecated by?

**Parameters**

| | | |
|---|---|---|
| in | *_contact* | The contact which was detected by a collision engine. |

#### 10.19.3.2 void gazebo::physics::Collision::FillMsg ( msgs::Collision & _msg )

Fill a collision message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | The message to fill with this collision's data. |

#### 10.19.3.3 virtual void gazebo::physics::Collision::Fini ( ) [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Entity** (p. 382).

Reimplemented in **gazebo::physics::DARTCollision** (p. 288).

#### 10.19.3.4 virtual math::Box gazebo::physics::Collision::GetBoundingBox ( ) const [pure virtual]

Get the bounding box for this collision.

**Returns**

The bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 382).

Implemented in **gazebo::physics::DARTCollision** (p. 288), and **gazebo::physics::SimbodyCollision** (p. 874).

#### 10.19.3.5 bool gazebo::physics::Collision::GetContactsEnabled ( ) const

Return true of contacts are on.

Deprecated by?

**Returns**

True of contact are on.

#### 10.19.3.6 float gazebo::physics::Collision::GetLaserRetro ( ) const

Get the laser retro reflectiveness.

**Returns**

The laser retro value.

#### 10.19.3.7 LinkPtr gazebo::physics::Collision::GetLink ( ) const

Get the link this collision belongs to.

**Returns**

The parent **Link** (p. 542).

#### 10.19.3.8 virtual int gazebo::physics::Collision::GetMaxContacts ( ) [virtual]

returns number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 706)) if specified.

**Returns**

max num contacts allowed for this collision.

#### 10.19.3.9 ModelPtr gazebo::physics::Collision::GetModel ( ) const

Get the model this collision belongs to.

**Returns**

The parent model.

#### 10.19.3.10 virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the collision.

**Returns**

The angular acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.19.3.11 virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the collision.

**Returns**

The angular velocity of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.19.3.12 virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the collision.

**Returns**

The linear acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.19.3.13 virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the collision.

**Returns**

The linear velocity relative to the parent model.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.19.3.14 ShapePtr gazebo::physics::Collision::GetShape ( ) const

Get the collision shape.

**Returns**

The collision shape.

#### 10.19.3.15 unsigned int gazebo::physics::Collision::GetShapeType ( )

Get the shape type.

**Returns**

The shape type.

**See Also**

**EntityType** (p. 163)

#### 10.19.3.16 CollisionState gazebo::physics::Collision::GetState ( )

Get the collision state.

**Returns**

The collision state.

#### 10.19.3.17 SurfaceParamsPtr gazebo::physics::Collision::GetSurface ( ) const [inline]

Get the surface parameters.

**Returns**

The surface parameters.

#### 10.19.3.18 virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the collision in the world frame.

**Returns**

The angular acceleration of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.19.3.19 virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the collision in the world frame.

**Returns**

The angular velocity of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.19.3.20 virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the collision in the world frame.

**Returns**

The linear acceleration of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.19.3.21 virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the collision in the world frame.

**Returns**

The linear velocity of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.19.3.22 virtual void gazebo::physics::Collision::Init ( ) [virtual]

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 167).

Reimplemented in **gazebo::physics::DARTCollision** (p. 289).

#### 10.19.3.23 bool gazebo::physics::Collision::IsPlaceable ( ) const

Return whether this collision is movable.

Example on an immovable object is a ray.

**Returns**

True if the object is immovable.

#### 10.19.3.24 virtual void gazebo::physics::Collision::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF to load from. |

Reimplemented from **gazebo::physics::Entity** (p. 386).

Reimplemented in **gazebo::physics::SimbodyCollision** (p. 874), and **gazebo::physics::DARTCollision** (p. 289).

#### 10.19.3.25 void gazebo::physics::Collision::ProcessMsg ( const msgs::Collision & *_msg* )

Update parameters from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to update from. |

#### 10.19.3.26 virtual void gazebo::physics::Collision::SetCategoryBits ( unsigned int *_bits* ) [pure virtual]

Set the category bits, used during collision detection.

**Parameters**

| | | |
|---|---|---|
| in | *_bits* | The bits to set. |

Implemented in **gazebo::physics::DARTCollision** (p. 290), and **gazebo::physics::SimbodyCollision** (p. 875).

#### 10.19.3.27 virtual void gazebo::physics::Collision::SetCollideBits ( unsigned int *_bits* ) [pure virtual]

Set the collide bits, used during collision detection.

**Parameters**

| | | |
|---|---|---|
| in | *_bits* | The bits to set. |

Implemented in **gazebo::physics::DARTCollision** (p. 290), and **gazebo::physics::SimbodyCollision** (p. 875).

#### 10.19.3.28 void gazebo::physics::Collision::SetCollision ( bool *_placeable* )

Set the encapsulated collsion object.

**Parameters**

| | | |
|---|---|---|
| in | *_placeable* | True to make the object movable. |

#### 10.19.3.29 void gazebo::physics::Collision::SetContactsEnabled ( bool *_enable* )

Turn contact recording on or off.

Deprecated by?

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | True to enable collision contacts. |

#### 10.19.3.30 void gazebo::physics::Collision::SetLaserRetro ( float *_retro* )

Set the laser retro reflectiveness.

**Parameters**

| | | |
|---|---|---|
| in | *_retro* | The laser retro value. |

#### 10.19.3.31 virtual void gazebo::physics::Collision::SetMaxContacts ( double *_maxContacts* ) [virtual]

Number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 706)) if specified.

**Parameters**

| | | |
|---|---|---|
| in | *_maxContacts* | max num contacts allowed for this collision. |

#### 10.19.3.32 void gazebo::physics::Collision::SetScale ( const **math::Vector3** & *_scale* )

Set the scale of the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the collision to. |

#### 10.19.3.33 void gazebo::physics::Collision::SetShape ( **ShapePtr** *_shape* )

Set the shape for this collision.

**Parameters**

| | | |
|---|---|---|
| in | *_shape* | The shape for this collision object. |

#### 10.19.3.34 void gazebo::physics::Collision::SetState ( const CollisionState & *_state* )

Set the current collision state.

**Parameters**

| | | |
|---|---|---|
| in | *The* | collision state. |

#### 10.19.3.35 virtual void gazebo::physics::Collision::UpdateParameters ( sdf::ElementPtr *_sdf* ) [virtual]

Update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to update from. |

Reimplemented from **gazebo::physics::Entity** (p. 389).

### 10.19.4 Member Data Documentation

#### 10.19.4.1 LinkPtr gazebo::physics::Collision::link [protected]

The link this collision belongs to.

#### 10.19.4.2 bool gazebo::physics::Collision::placeable [protected]

Flag for placeable.

#### 10.19.4.3 ShapePtr gazebo::physics::Collision::shape [protected]

Pointer to **physics::Shape** (p. 861).

The documentation for this class was generated from the following file:

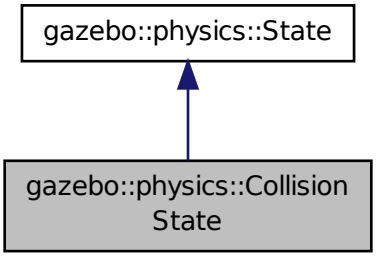
- **Collision.hh**

## 10.20 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 220) object.

`#include <physics/phyiscs.hh>`

Inheritance diagram for gazebo::physics::CollisionState:




## Public Member Functions

- **CollisionState** ()

  *Default constructor.*
- **CollisionState** (const **CollisionPtr** _collision)

  *Constructor.*
- **CollisionState** (const sdf::ElementPtr _sdf)

  *Constructor.*
- virtual ~**CollisionState** ()

  *Destructor.*
- void **FillSDF** (sdf::ElementPtr _sdf)

  *Populate a state SDF element with data from the object.*
- const **math::Pose** & **GetPose** () const

  *Get the **Collision** (p. 220) pose.*
- bool **IsZero** () const

  *Return true if the values in the state are zero.*
- virtual void **Load** (const sdf::ElementPtr _elem)

  *Load state from SDF element.*
- **CollisionState operator+** (const **CollisionState** &_state) const

  *Addition operator.*
- **CollisionState operator-** (const **CollisionState** &_state) const

  *Subtraction operator.*
- **CollisionState** & **operator=** (const **CollisionState** &_state)

  *Assignment operator.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::CollisionState** &_state)

  *Stream insertion operator.*

### Additional Inherited Members

## 10.20.1 Detailed Description

Store state information of a **physics::Collision** (p. 220) object.

This class captures the entire state of a **Collision** (p. 220) at one specific time during a simulation run.

**State** (p. 998) of a **Collision** (p. 220) is its Pose.

## 10.20.2 Constructor & Destructor Documentation

#### 10.20.2.1 gazebo::physics::CollisionState::CollisionState ( )

Default constructor.

#### 10.20.2.2 gazebo::physics::CollisionState::CollisionState ( const CollisionPtr *_collision* ) [explicit]

Constructor.

Build a **CollisionState** (p. 229) from an existing **Collision** (p. 220).

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the **Link** (p. 542) from which to gather state info. |

#### 10.20.2.3 gazebo::physics::CollisionState::CollisionState ( const sdf::ElementPtr *_sdf* ) [explicit]

Constructor.

Build a **CollisionState** (p. 229) from SDF data

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF data to load a collision state from. |

#### 10.20.2.4 virtual gazebo::physics::CollisionState::~CollisionState ( ) [virtual]

Destructor.

## 10.20.3 Member Function Documentation

#### 10.20.3.1 void gazebo::physics::CollisionState::FillSDF ( sdf::ElementPtr *_sdf* )

Populate a state SDF element with data from the object.

**Parameters**

| | | |
|---|---|---|
| out | *_sdf* | SDF element to populate. |

#### 10.20.3.2 const math::Pose& gazebo::physics::CollisionState::GetPose ( ) const

Get the **Collision** (p. 220) pose.

**Returns**

The pose of the **CollisionState** (p. 229)

#### 10.20.3.3 bool gazebo::physics::CollisionState::IsZero ( ) const

Return true if the values in the state are zero.

**Returns**

True if the values in the state are zero.

#### 10.20.3.4 virtual void gazebo::physics::CollisionState::Load ( const sdf::ElementPtr *_elem* ) [virtual]

Load state from SDF element.

Load **CollisionState** (p. 229) information from stored data in and SDF::Element

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | Pointer to the SDF::Element containing state info. |

Reimplemented from **gazebo::physics::State** (p. 1000).

#### 10.20.3.5 CollisionState gazebo::physics::CollisionState::operator+ ( const CollisionState & *_state* ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to add. |

**Returns**

The resulting state.

#### 10.20.3.6 CollisionState gazebo::physics::CollisionState::operator- ( const CollisionState & *_state* ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to substract. |

**Returns**

The resulting state.

#### 10.20.3.7 CollisionState& gazebo::physics::CollisionState::operator= ( const CollisionState & *_state* )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **State** (p. 998) value |

**Returns**

Reference to this

### 10.20.4 Friends And Related Function Documentation

#### 10.20.4.1 std::ostream& operator<< ( std::ostream & *_out*, const gazebo::physics::CollisionState & *_state* ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_state* | **Collision** (p. 220) state to output |

**Returns**

the stream

The documentation for this class was generated from the following file:

- **CollisionState.hh**

## 10.21 gazebo::common::Color Class Reference

Defines a color.

`#include <common/common.hh>`

### Public Types

- typedef unsigned int **ABGR**
- typedef unsigned int **ARGB**
- typedef unsigned int **BGRA**
- typedef unsigned int **RGBA**

## Public Member Functions

- **Color** ()

  *Constructor.*
- **Color** (float _r, float _g, float _b, float _a=1.0)

  *Constructor.*
- **Color** (const **Color** &_clr)

  *Copy Constructor.*
- virtual ~**Color** ()

  *Destructor.*
- **ABGR GetAsABGR** () const

  *Get as uint32 ABGR packed value.*
- **ARGB GetAsARGB** () const

  *Get as uint32 ARGB packed value.*
- **BGRA GetAsBGRA** () const

  *Get as uint32 BGRA packed value.*
- **math::Vector3 GetAsHSV** () const

  *Get the color in HSV colorspace.*
- **RGBA GetAsRGBA** () const

  *Get as uint32 RGBA packed value.*
- **math::Vector3 GetAsYUV** () const

  *Get the color in YUV colorspace.*
- bool **operator!=** (const **Color** &_pt) const

  *Inequality operator.*
- const **Color operator**∗ (const **Color** &_pt) const

  *Multiplication operator.*
- const **Color operator**∗ (const float &_v) const

  *Multiply all color components by _v.*
- const **Color** & **operator∗=** (const **Color** &_pt)

  *Multiplication equal operator.*
- **Color operator+** (const **Color** &_pt) const

  *Addition operator (this + _pt)*
- **Color operator+** (const float &_v) const

  *Add _v to all color components.*
- const **Color** & **operator+=** (const **Color** &_pt)

  *Addition equal operator.*
- **Color operator-** (const **Color** &_pt) const

  *Subtraction operator.*
- **Color operator-** (const float &_v) const

  *Subtract _v from all color components.*
- const **Color** & **operator-=** (const **Color** &_pt)

  *Subtraction equal operator.*
- const **Color operator**/ (const **Color** &_pt) const

  *Division operator.*
- const **Color operator**/ (const float &_v) const

  *Divide all color component by _v.*
- const **Color** & **operator/=** (const **Color** &_pt)

*Division equal operator.*

- **Color** & **operator=** (const **Color** &_pt)

  *Equal operator.*
- bool **operator==** (const **Color** &_pt) const

  *Equality operator.*
- float **operator[]** (unsigned int _index)

  *Array index operator.*
- void **Reset** ()

  *Reset the color to default values.*
- void **Set** (float _r=1, float _g=1, float _b=1, float _a=1)

  *Set the contents of the vector.*
- void **SetFromABGR** (const **ABGR** _v)

  *Set from uint32 ABGR packed value.*
- void **SetFromARGB** (const **ARGB** _v)

  *Set from uint32 ARGB packed value.*
- void **SetFromBGRA** (const **BGRA** _v)

  *Set from uint32 BGRA packed value.*
- void **SetFromHSV** (float _h, float _s, float _v)

  *Set a color based on HSV values.*
- void **SetFromRGBA** (const **RGBA** _v)

  *Set from uint32 RGBA packed value.*
- void **SetFromYUV** (float _y, float _u, float _v)

  *Set from yuv.*

## Public Attributes

- float **a**
- float **b**
- float **g**
- float **r**

## Static Public Attributes

- static const **Color Black**

  *(0, 0, 0)*
- static const **Color Blue**

  *(0, 0, 1)*
- static const **Color Green**

  *(0, 1, 0)*
- static const **Color Purple**

  *(1, 0, 1)*
- static const **Color Red**

  *(1, 0, 0)*
- static const **Color White**

  *(1, 1, 1)*
- static const **Color Yellow**

  *(1, 1, 0)*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **Color** &_pt)

  *Stream insertion operator.*
- std::istream & **operator**>> (std::istream &_in, **Color** &_pt)

  *Stream insertion operator.*

### 10.21.1 Detailed Description

Defines a color.

### 10.21.2 Member Typedef Documentation

#### 10.21.2.1 typedef unsigned int gazebo::common::Color::ABGR

#### 10.21.2.2 typedef unsigned int gazebo::common::Color::ARGB

#### 10.21.2.3 typedef unsigned int gazebo::common::Color::BGRA

#### 10.21.2.4 typedef unsigned int gazebo::common::Color::RGBA

### 10.21.3 Constructor & Destructor Documentation

#### 10.21.3.1 gazebo::common::Color::Color ( )

Constructor.

#### 10.21.3.2 gazebo::common::Color::Color ( float *_r*, float *_g*, float *_b*, float *_a* = `1.0` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_r* | Red value (range 0 to 1) |
| in | *_g* | Green value (range 0 to 1 |
| in | *_b* | Blue value (range 0 to 1 |
| in | *_a* | Alpha value (0=transparent, 1=opaque) |

#### 10.21.3.3 gazebo::common::Color::Color ( const **Color** & *_clr* )

Copy Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | **Color** (p. 233) to copy |

#### 10.21.3.4 virtual gazebo::common::Color::~Color ( ) [virtual]

Destructor.

### 10.21.4 Member Function Documentation

#### 10.21.4.1 ABGR gazebo::common::Color::GetAsABGR ( ) const

Get as uint32 ABGR packed value.

**Returns**

the color

#### 10.21.4.2 ARGB gazebo::common::Color::GetAsARGB ( ) const

Get as uint32 ARGB packed value.

**Returns**

the color

#### 10.21.4.3 BGRA gazebo::common::Color::GetAsBGRA ( ) const

Get as uint32 BGRA packed value.

**Returns**

the color

#### 10.21.4.4 math::Vector3 gazebo::common::Color::GetAsHSV ( ) const

Get the color in HSV colorspace.

**Returns**

HSV values in a **math::Vector3** (p. 1091) format

#### 10.21.4.5 RGBA gazebo::common::Color::GetAsRGBA ( ) const

Get as uint32 RGBA packed value.

**Returns**

the color

#### 10.21.4.6 math::Vector3 gazebo::common::Color::GetAsYUV ( ) const

Get the color in YUV colorspace.

**Returns**

the YUV color

#### 10.21.4.7 bool gazebo::common::Color::operator!= ( const Color & _pt ) const

Inequality operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The color to check for inequality |

**Returns**

True if the this color does not equal _pt

#### 10.21.4.8 const Color gazebo::common::Color::operator* ( const Color & _pt ) const

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The color to muliply by |

**Returns**

The resulting color

#### 10.21.4.9 const Color gazebo::common::Color::operator* ( const float & _v ) const

Multiply all color components by _v.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | The value to multiply by |

**Returns**

The resulting color

#### 10.21.4.10 const Color& gazebo::common::Color::operator*= ( const Color & _pt )

Multiplication equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The color to muliply by |

**Returns**

The resulting color

#### 10.21.4.11 Color gazebo::common::Color::operator+ ( const Color & _pt ) const

Addition operator (this + _pt)

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | **Color** (p. 233) to add |

**Returns**

The resulting color

#### 10.21.4.12 Color gazebo::common::Color::operator+ ( const float & _v ) const

Add _v to all color components.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | Value to add to each color component |

**Returns**

The resulting color

#### 10.21.4.13 const Color& gazebo::common::Color::operator+= ( const Color & _pt )

Addition equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | **Color** (p. 233) to add |

**Returns**

The resulting color

#### 10.21.4.14 Color gazebo::common::Color::operator- ( const Color & _pt ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The color to substract |

**Returns**

The resulting color

#### 10.21.4.15 Color gazebo::common::Color::operator- ( const float & _v ) const

Subtract _v from all color components.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | Value to subtract |

**Returns**

The resulting color

#### 10.21.4.16 const Color& gazebo::common::Color::operator-= ( const Color & _pt )

Subtraction equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | **Color** (p. 233) to subtract |

**Returns**

The resulting color

#### 10.21.4.17 const Color gazebo::common::Color::operator/ ( const Color & _pt ) const

Division operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | **Color** (p. 233) to divide by |

**Returns**

The resulting color

#### 10.21.4.18 const Color gazebo::common::Color::operator/ ( const float & _v ) const

Divide all color component by _v.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | The value to divide by |

**Returns**

The resulting color

#### 10.21.4.19 const Color& gazebo::common::Color::operator/= ( const Color & _pt )

Division equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | **Color** (p. 233) to divide by |

**Returns**

The resulting color

#### 10.21.4.20 Color& gazebo::common::Color::operator= ( const Color & _pt )

Equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | **Color** (p. 233) to copy |

**Returns**

Reference to this color

#### 10.21.4.21 bool gazebo::common::Color::operator== ( const Color & _pt ) const

Equality operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The color to check for equality |

**Returns**

True if the this color equals _pt

#### 10.21.4.22 float gazebo::common::Color::operator[] ( unsigned int _index )

Array index operator.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | **Color** (p. 233) component index(0=red, 1=green, 2=blue) |

**Returns**

r, g, b, or a when _index is 0, 1, 2 or 3

#### 10.21.4.23 void gazebo::common::Color::Reset ( )

Reset the color to default values.

#### 10.21.4.24 void gazebo::common::Color::Set ( float *_r* = 1, float *_g* = 1, float *_b* = 1, float *_a* = 1 )

Set the contents of the vector.

**Parameters**

| | | |
|---|---|---|
| in | *_r* | Red value (range 0 to 1) |
| in | *_g* | Green value (range 0 to 1) |
| in | *_b* | Blue value (range 0 to 1) |
| in | *_a* | Alpha value (0=transparent, 1=opaque) |

#### 10.21.4.25 void gazebo::common::Color::SetFromABGR ( const **ABGR** *_v* )

Set from uint32 ABGR packed value.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the new color |

#### 10.21.4.26 void gazebo::common::Color::SetFromARGB ( const **ARGB** *_v* )

Set from uint32 ARGB packed value.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the new color |

#### 10.21.4.27 void gazebo::common::Color::SetFromBGRA ( const **BGRA** *_v* )

Set from uint32 BGRA packed value.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the new color |

#### 10.21.4.28 void gazebo::common::Color::SetFromHSV ( float _h, float _s, float _v )

Set a color based on HSV values.

**Parameters**

| | | |
|---|---|---|
| in | _h | Hue(0..360) |
| in | _s | Saturation(0..1) |
| in | _v | Value(0..1) |

#### 10.21.4.29 void gazebo::common::Color::SetFromRGBA ( const RGBA _v )

Set from uint32 RGBA packed value.

**Parameters**

| | | |
|---|---|---|
| in | _v | the new color |

#### 10.21.4.30 void gazebo::common::Color::SetFromYUV ( float _y, float _u, float _v )

Set from yuv.

**Parameters**

| | | |
|---|---|---|
| in | _y | value |
| in | _u | value |
| in | _v | value |

### 10.21.5 Friends And Related Function Documentation

#### 10.21.5.1 std::ostream& operator<< ( std::ostream & _out, const Color & _pt ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | _out | the output stream |
| in | _pt | the color |

**Returns**

the output stream

#### 10.21.5.2 std::istream& operator>> ( std::istream & _in, Color & _pt ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | _in | the input stream |
| in | _pt | |

### 10.21.6 Member Data Documentation

#### 10.21.6.1 float gazebo::common::Color::a

#### 10.21.6.2 float gazebo::common::Color::b

#### 10.21.6.3 const Color gazebo::common::Color::Black [static]

(0, 0, 0)

#### 10.21.6.4 const Color gazebo::common::Color::Blue [static]

(0, 0, 1)

#### 10.21.6.5 float gazebo::common::Color::g

#### 10.21.6.6 const Color gazebo::common::Color::Green [static]

(0, 1, 0)

#### 10.21.6.7 const Color gazebo::common::Color::Purple [static]

(1, 0, 1)

#### 10.21.6.8 float gazebo::common::Color::r

#### 10.21.6.9 const Color gazebo::common::Color::Red [static]

(1, 0, 0)

#### 10.21.6.10 const Color gazebo::common::Color::White [static]

(1, 1, 1)

#### 10.21.6.11 const Color gazebo::common::Color::Yellow [static]

(1, 1, 0)

The documentation for this class was generated from the following file:

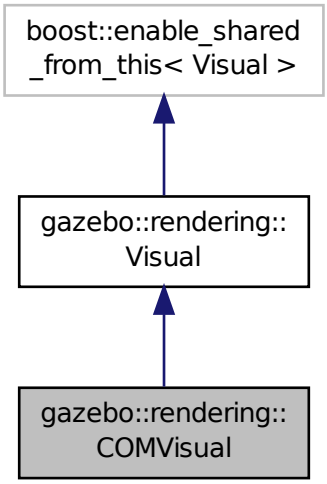
- **Color.hh**

## 10.22 gazebo::rendering::COMVisual Class Reference

Basic Center of Mass visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::COMVisual:




## Public Member Functions

- **COMVisual** (const std::string &_name, **VisualPtr** _vis)

  *Constructor.*
- virtual ~**COMVisual** ()

  *Destructor.*
- virtual void **Load** (sdf::ElementPtr _elem)

  *Load the **Visual** (p. 1121) from an SDF pointer.*
- virtual void **Load** (ConstLinkPtr &_msg)

  *Load from a message.*

## Additional Inherited Members

### 10.22.1 Detailed Description

Basic Center of Mass visualization.

### 10.22.2 Constructor & Destructor Documentation

#### 10.22.2.1 gazebo::rendering::COMVisual::COMVisual ( const std::string & *_name*, VisualPtr *_vis* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the **Visual** (p. 1121) |
| in | *_vis* | Parent **Visual** (p. 1121) |

#### 10.22.2.2 virtual gazebo::rendering::COMVisual::~COMVisual ( ) [virtual]

Destructor.

### 10.22.3 Member Function Documentation

#### 10.22.3.1 virtual void gazebo::rendering::COMVisual::Load ( sdf::ElementPtr *_elem* ) [virtual]

Load the **Visual** (p. 1121) from an SDF pointer.

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | SDF Element pointer |

#### 10.22.3.2 virtual void gazebo::rendering::COMVisual::Load ( ConstLinkPtr & *_msg* ) [virtual]

Load from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Pointer to the message |

The documentation for this class was generated from the following file:

- **COMVisual.hh**

## 10.23 gazebo::event::Connection Class Reference

A class that encapsulates a connection.

`#include <Event.hh>`

### Public Member Functions

- **Connection** ()

  *Constructor.*
- **Connection** (**Event** *_e, int _i)

  *Constructor.*
- ~**Connection** ()

  *Destructor.*
- int **GetId** () const

  *Get the id of this connection.*

### 10.23.1 Detailed Description

A class that encapsulates a connection.

### 10.23.2 Constructor & Destructor Documentation

#### 10.23.2.1 gazebo::event::Connection::Connection ( ) [inline]

Constructor.

#### 10.23.2.2 gazebo::event::Connection::Connection ( Event * _e, int _i )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_e* | **Event** (p. 390) pointer to connect with |
| in | *_i* | Unique id |

#### 10.23.2.3 gazebo::event::Connection::~Connection ( )

Destructor.

### 10.23.3 Member Function Documentation

#### 10.23.3.1 int gazebo::event::Connection::GetId ( ) const

Get the id of this connection.

**Returns**

The id of this connection

The documentation for this class was generated from the following file:

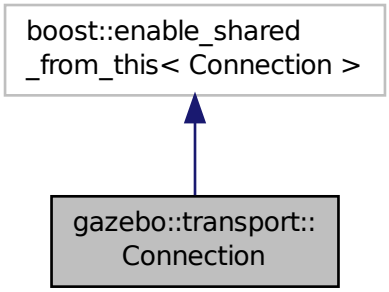
- **Event.hh**

## 10.24 gazebo::transport::Connection Class Reference

Single TCP/IP connection manager.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::Connection:




## Public Types

- typedef boost::function< void(const **ConnectionPtr** &)> **AcceptCallback**

  *The signature of a connection accept callback.*
- typedef boost::function< void(const std::string &_data)> **ReadCallback**

  *The signature of a connection read callback.*

## Public Member Functions

- **Connection** ()

  *Constructor.*
- virtual ~**Connection** ()

  *Destructor.*
- template<typename Handler > void **AsyncRead** (Handler _handler)

  *Peform an asyncronous read param[in] _handler Callback to invoke on received data.*
- void **Cancel** ()

  *Cancel all async operations on an open socket.*
- bool **Connect** (const std::string &_host, unsigned int _port)

  *Connect to a remote host.*
- **event::ConnectionPtr ConnectToShutdown** (boost::function< void()> _subscriber)

  *Register a function to be called when the connection is shut down.*
- void **DisconnectShutdown** (**event::ConnectionPtr** _subscriber)

  *Unregister a function to be called when the connection is shut down.*
- void **EnqueueMsg** (const std::string &_buffer, boost::function< void(uint32_t)> _cb, uint32_t _id, bool _force=false)

  *Write data to the socket.*
- void **EnqueueMsg** (const std::string &_buffer, bool _force=false)

*Write data to the socket.*

- unsigned int **GetId** () const

  *Get the ID of the connection.*

- std::string **GetIPWhiteList** () const

  *Get the IP white list, from GAZEBO_IP_WHITE_LIST environment variable.*

- std::string **GetLocalAddress** () const

  *Get the local address of this connection.*

- unsigned int **GetLocalPort** () const

  *Get the port of this connection.*

- std::string **GetLocalURI** () const

  *Get the local URI.*

- std::string **GetRemoteAddress** () const

  *Get the remote address.*

- std::string **GetRemoteHostname** () const

  *Get the remote hostname.*

- unsigned int **GetRemotePort** () const

  *Get the remote port number.*

- std::string **GetRemoteURI** () const

  *Get the remote URI.*

- bool **IsOpen** () const

  *Is the connection open?*

- void **Listen** (unsigned int _port, const **AcceptCallback** &_acceptCB)

  *Start a server that listens on a port.*

- void **ProcessWriteQueue** (bool _blocking=false)

  *Handle on-write callbacks.*

- bool **Read** (std::string &_data)

  *Read data from the socket.*

- void **Shutdown** ()

  *Shutdown the socket.*

- void **StartRead** (const **ReadCallback** &_cb)

  *Start a thread that reads from the connection and passes new message to the ReadCallback.*

- void **StopRead** ()

  *Stop the read loop.*

## Static Public Member Functions

- static std::string **GetLocalHostname** ()

  *Get the local hostname.*

- static bool **ValidateIP** (const std::string &_ip)

  *Return true if the _ip is a valid.*

### 10.24.1 Detailed Description

Single TCP/IP connection manager.

### 10.24.2 Member Typedef Documentation

#### 10.24.2.1 typedef boost::function<void(const ConnectionPtr&)> gazebo::transport::Connection::AcceptCallback

The signature of a connection accept callback.

#### 10.24.2.2 typedef boost::function<void(const std::string &_data)> gazebo::transport::Connection::ReadCallback

The signature of a connection read callback.

### 10.24.3 Constructor & Destructor Documentation

#### 10.24.3.1 gazebo::transport::Connection::Connection ( )

Constructor.

#### 10.24.3.2 virtual gazebo::transport::Connection::~Connection ( ) [virtual]

Destructor.

### 10.24.4 Member Function Documentation

#### 10.24.4.1 template<typename Handler > void gazebo::transport::Connection::AsyncRead ( Handler *_handler* ) [inline]

Peform an asyncronous read param[in] _handler Callback to invoke on received data.

References gzerr, HEADER_LENGTH, and IsOpen().

#### 10.24.4.2 void gazebo::transport::Connection::Cancel ( )

Cancel all async operations on an open socket.

#### 10.24.4.3 bool gazebo::transport::Connection::Connect ( const std::string & *_host*, unsigned int *_port* )

Connect to a remote host.

**Parameters**

| | | |
|---|---|---|
| in | *_host* | The host to connect to |
| in | *_port* | The port to connect to |

**Returns**

true if connection succeeded, false otherwise

#### 10.24.4.4 event::ConnectionPtr gazebo::transport::Connection::ConnectToShutdown ( boost::function< void()> *_subscriber* ) [inline]

Register a function to be called when the connection is shut down.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Function to be called |

**Returns**

Handle that can be used to unregister the function

References gazebo::event::EventT< T >::Connect().

#### 10.24.4.5 void gazebo::transport::Connection::DisconnectShutdown ( event::ConnectionPtr *_subscriber* ) [inline]

Unregister a function to be called when the connection is shut down.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Handle previously returned by **ConnectToShutdown()** (p. 250) |

References gazebo::event::EventT< T >::Disconnect().

#### 10.24.4.6 void gazebo::transport::Connection::EnqueueMsg ( const std::string & *_buffer,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id,* bool *_force* = false )

Write data to the socket.

**Parameters**

| | | |
|---|---|---|
| in | *_buffer* | Data to write |
| in | *_force* | If true, block until the data has been written to the socket, otherwise just enqueue the data for asynchronous write |
| in | *_cb* | If non-null, callback to be invoked after transmission is complete. |
| in | *_id* | ID associated with the message data. |

#### 10.24.4.7 void gazebo::transport::Connection::EnqueueMsg ( const std::string & *_buffer,* bool *_force* = false )

Write data to the socket.

**Parameters**

| | | |
|---|---|---|
| in | *_buffer* | Data to write |
| in | *_force* | If true, block until the data has been written to the socket, otherwise just enqueue the data for asynchronous write |

#### 10.24.4.8 unsigned int gazebo::transport::Connection::GetId ( ) const

Get the ID of the connection.

**Returns**

The connection's unique ID.

#### 10.24.4.9 std::string gazebo::transport::Connection::GetIPWhiteList ( ) const

Get the IP white list, from GAZEBO_IP_WHITE_LIST environment variable.

**Returns**

GAZEBO_IP_WHITE_LIST

#### 10.24.4.10 std::string gazebo::transport::Connection::GetLocalAddress ( ) const

Get the local address of this connection.

**Returns**

The local address

#### 10.24.4.11 static std::string gazebo::transport::Connection::GetLocalHostname ( ) [static]

Get the local hostname.

**Returns**

The local hostname

#### 10.24.4.12 unsigned int gazebo::transport::Connection::GetLocalPort ( ) const

Get the port of this connection.

**Returns**

The local port

#### 10.24.4.13 std::string gazebo::transport::Connection::GetLocalURI ( ) const

Get the local URI.

**Returns**

The local URI

#### 10.24.4.14 std::string gazebo::transport::Connection::GetRemoteAddress ( ) const

Get the remote address.

**Returns**

The remote address

#### 10.24.4.15 std::string gazebo::transport::Connection::GetRemoteHostname ( ) const

Get the remote hostname.

**Returns**

The remote hostname

#### 10.24.4.16 unsigned int gazebo::transport::Connection::GetRemotePort ( ) const

Get the remote port number.

**Returns**

The remote port

#### 10.24.4.17 std::string gazebo::transport::Connection::GetRemoteURI ( ) const

Get the remote URI.

**Returns**

The remote URI

#### 10.24.4.18 bool gazebo::transport::Connection::IsOpen ( ) const

Is the connection open?

**Returns**

true if the connection is open; false otherwise

Referenced by AsyncRead().

#### 10.24.4.19 void gazebo::transport::Connection::Listen ( unsigned int *_port,* const **AcceptCallback** & *_acceptCB* )

Start a server that listens on a port.

**Parameters**

| | | |
|---|---|---|
| in | *_port* | The port to listen on |
| in | *_acceptCB* | The callback to invoke when a new connection has been accepted |

#### 10.24.4.20 void gazebo::transport::Connection::ProcessWriteQueue ( bool *_blocking* = `false` )

Handle on-write callbacks.

#### 10.24.4.21 bool gazebo::transport::Connection::Read ( std::string & _data )

Read data from the socket.

**Parameters**

| | | |
|---|---|---|
| out | _data | Destination for data that is read |

**Returns**

true if data was successfully read, false otherwise

#### 10.24.4.22 void gazebo::transport::Connection::Shutdown ( )

Shutdown the socket.

#### 10.24.4.23 void gazebo::transport::Connection::StartRead ( const ReadCallback & _cb )

Start a thread that reads from the connection and passes new message to the ReadCallback.

**Parameters**

| | | |
|---|---|---|
| in | _cb | The callback to invoke when a new message is received |

#### 10.24.4.24 void gazebo::transport::Connection::StopRead ( )

Stop the read loop.

#### 10.24.4.25 static bool gazebo::transport::Connection::ValidateIP ( const std::string & _ip ) [static]

Return true if the _ip is a valid.

**Parameters**

| | | |
|---|---|---|
| in | _ip | Dotted quad to validate. |

**Returns**

True if the _ip is a valid.

The documentation for this class was generated from the following file:

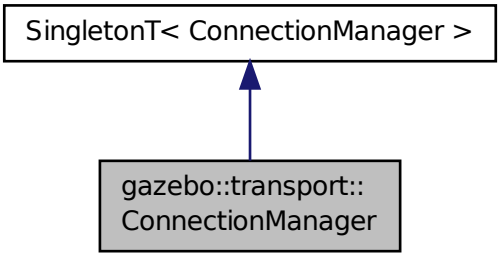
- **Connection.hh**

## 10.25 gazebo::transport::ConnectionManager Class Reference

Manager of connections.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::ConnectionManager:




## Public Member Functions

- void **Advertise** (const std::string &_topic, const std::string &_msgType)

  *Advertise a topic.*
- **ConnectionPtr ConnectToRemoteHost** (const std::string &_host, unsigned int _port)

  *Connect to a remote server.*
- void **Fini** ()

  *Finalize the connection manager.*
- void **GetAllPublishers** (std::list< msgs::Publish > &_publishers)

  *Explicitly update the publisher list.*
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  *Get all the topic namespaces.*
- bool **Init** (const std::string &_masterHost, unsigned int _masterPort)

  *Initialize the connection manager.*
- bool **IsRunning** () const

  *Is the manager running?*
- void **RegisterTopicNamespace** (const std::string &_name)

  *Register a new topic namespace.*
- void **RemoveConnection** (**ConnectionPtr** &_conn)

  *Remove a connection from the manager.*
- void **Run** ()

  *Run the connection manager loop.*
- void **Stop** ()

  *Stop the conneciton manager.*
- void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)

  *Subscribe to a topic.*
- void **TriggerUpdate** ()

  *Inform the connection manager that it needs an update.*
- void **Unadvertise** (const std::string &_topic)

  *Unadvertise a topic.*

- void **Unsubscribe** (const msgs::Subscribe &_sub)

  *Unsubscribe from a topic.*
- void **Unsubscribe** (const std::string &_topic, const std::string &_msgType)

  *Unsubscribe from a topic.*

### Protected Attributes

- std::vector< **event::ConnectionPtr** > **eventConnections**

### Additional Inherited Members

### 10.25.1 Detailed Description

Manager of connections.

### 10.25.2 Member Function Documentation

#### 10.25.2.1 void gazebo::transport::ConnectionManager::Advertise ( const std::string & *_topic,* const std::string & *_msgType* )

Advertise a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to advertise |
| in | *_msgType* | The type of the topic |

#### 10.25.2.2 ConnectionPtr gazebo::transport::ConnectionManager::ConnectToRemoteHost ( const std::string & *_host,* unsigned int *_port* )

Connect to a remote server.

**Parameters**

| | | |
|---|---|---|
| in | *_host* | Host to connect to |
| in | *_port* | Port to connect to |

**Returns**

Pointer to the connection; can be null (if connection failed)

#### 10.25.2.3 void gazebo::transport::ConnectionManager::Fini ( )

Finalize the connection manager.

#### 10.25.2.4 void gazebo::transport::ConnectionManager::GetAllPublishers ( std::list< msgs::Publish > & *_publishers* )

Explicitly update the publisher list.

**Parameters**

| | | |
|---|---|---|
| out | *_publishers* | The updated list of publishers is written here |

#### 10.25.2.5 void gazebo::transport::ConnectionManager::GetTopicNamespaces ( std::list< std::string > & *_namespaces* )

Get all the topic namespaces.

**Parameters**

| | | |
|---|---|---|
| out | *_namespaces* | The list of namespace is written here |

#### 10.25.2.6 bool gazebo::transport::ConnectionManager::Init ( const std::string & *_masterHost,* unsigned int *_masterPort* )

Initialize the connection manager.

**Parameters**

| | | |
|---|---|---|
| in | *_masterHost* | Host where the master is running |
| in | *_masterPort* | Port where the master is running |

**Returns**

true if initialization succeeded, false otherwise

#### 10.25.2.7 bool gazebo::transport::ConnectionManager::IsRunning ( ) const

Is the manager running?

**Returns**

true if running, false otherwise

#### 10.25.2.8 void gazebo::transport::ConnectionManager::RegisterTopicNamespace ( const std::string & *_name* )

Register a new topic namespace.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the topic namespace to be registered |

#### 10.25.2.9 void gazebo::transport::ConnectionManager::RemoveConnection ( ConnectionPtr & *_conn* )

Remove a connection from the manager.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | The connection to be removed |

#### 10.25.2.10 void gazebo::transport::ConnectionManager::Run ( )

Run the connection manager loop.

Does not return until stopped.

#### 10.25.2.11 void gazebo::transport::ConnectionManager::Stop ( )

Stop the conneciton manager.

#### 10.25.2.12 void gazebo::transport::ConnectionManager::Subscribe ( const std::string & *_topic,* const std::string & *_msgType,* bool *_latching* )

Subscribe to a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to subscribe to |
| in | *_msgType* | The type of the topic |
| in | *_latching* | If true, latch the latest incoming message; otherwise don't |

#### 10.25.2.13 void gazebo::transport::ConnectionManager::TriggerUpdate ( )

Inform the connection manager that it needs an update.

#### 10.25.2.14 void gazebo::transport::ConnectionManager::Unadvertise ( const std::string & *_topic* )

Unadvertise a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to unadvertise |

#### 10.25.2.15 void gazebo::transport::ConnectionManager::Unsubscribe ( const msgs::Subscribe & *_sub* )

Unsubscribe from a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_sub* | A subscription object |

#### 10.25.2.16 void gazebo::transport::ConnectionManager::Unsubscribe ( const std::string & *_topic,* const std::string & *_msgType* )

Unsubscribe from a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to unsubscribe from |
| in | *_msgType* | The type of the topic |

### 10.25.3 Member Data Documentation

#### 10.25.3.1 std::vector<event::ConnectionPtr> gazebo::transport::ConnectionManager::eventConnections [protected]

The documentation for this class was generated from the following file:

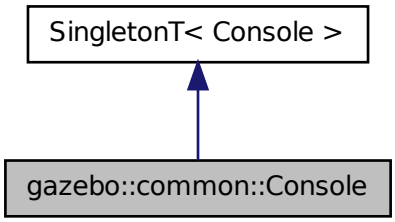
- **ConnectionManager.hh**

## 10.26 gazebo::common::Console Class Reference

Message, error, warning functionality.

`#include <common/commom.hh>`

Inheritance diagram for gazebo::common::Console:




### Public Member Functions

- std::ostream & **ColorErr** (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)

  *Use this to output an error to the terminal.*
- std::ostream & **ColorMsg** (const std::string &_lbl, int _color)

  *Use this to output a colored message to the terminal.*
- bool **GetQuiet** () const

  *Get whether quiet output is set.*
- void **Init** (const std::string &_logFilename)

  *Load the message parameters.*
- bool **IsInitialized** () const

  *Return true if Init has been called.*
- std::ofstream & **Log** ()

  *Use this to output a colored message to the terminal.*
- void **SetQuiet** (bool _q)

  *Set quiet output.*

### Additional Inherited Members

#### 10.26.1 Detailed Description

Message, error, warning functionality.

The documentation for this class was generated from the following file:

- **Console.hh**

## 10.27 gazebo::physics::Contact Class Reference

A contact between two collisions.

`#include <physics/physics.hh>`

### Public Member Functions

- **Contact** ()
  *Constructor.*
- **Contact** (const **Contact** &_contact)
  *Copy constructor.*
- virtual ~**Contact** ()
  *Destructor.*
- std::string **DebugString** () const
  *Produce a debug string.*
- void **FillMsg** (msgs::Contact &_msg) const
  *Populate a msgs::Contact with data from this.*
- **Contact** & **operator=** (const **Contact** &_contact)
  *Operator =.*
- **Contact** & **operator=** (const msgs::Contact &_contact)
  *Operator =.*
- void **Reset** ()
  *Reset to default values.*

### Public Attributes

- **Collision** * **collision1**
  *Pointer to the first collision object.*
- **Collision** * **collision2**
  *Pointer to the second collision object.*
- int **count**
  *Length of all the arrays.*
- double **depths** [32]
  *Array of contact depths.*
- **math::Vector3 normals** [32]
  *Array of force normals.*
- **math::Vector3 positions** [32]

*Array of force positions.*

- **common::Time time**

  *Time at which the contact occurred.*

- **WorldPtr world**

  ***World*** (p. *1157*) *in which the contact occurred.*

- **JointWrench wrench** [32]

  *Array of forces for the contact.*

### 10.27.1 Detailed Description

A contact between two collisions.

Each contact can consist of a number of contact points

### 10.27.2 Constructor & Destructor Documentation

#### 10.27.2.1 gazebo::physics::Contact::Contact ( )

Constructor.

#### 10.27.2.2 gazebo::physics::Contact::Contact ( const Contact & *_contact* )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_contact* | **Contact** (p. 260) to copy. |

#### 10.27.2.3 virtual gazebo::physics::Contact::~Contact ( ) [virtual]

Destructor.

### 10.27.3 Member Function Documentation

#### 10.27.3.1 std::string gazebo::physics::Contact::DebugString ( ) const

Produce a debug string.

**Returns**

A string that contains the values of the contact.

#### 10.27.3.2 void gazebo::physics::Contact::FillMsg ( msgs::Contact & *_msg* ) const

Populate a msgs::Contact with data from this.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | **Contact** (p. 260) message the will hold the data. |

#### 10.27.3.3 Contact& gazebo::physics::Contact::operator= ( const Contact & *_contact* )

Operator =.

**Parameters**

| | | |
|---|---|---|
| in | *_contact* | **Contact** (p. 260) to copy. |

**Returns**

Reference to this contact

#### 10.27.3.4 Contact& gazebo::physics::Contact::operator= ( const msgs::Contact & *_contact* )

Operator =.

**Parameters**

| | | |
|---|---|---|
| in | *_contact* | msgs::Contact to copy. |

**Returns**

Reference to this contact

#### 10.27.3.5 void gazebo::physics::Contact::Reset ( )

Reset to default values.

### 10.27.4 Member Data Documentation

#### 10.27.4.1 Collision* gazebo::physics::Contact::collision1

Pointer to the first collision object.

#### 10.27.4.2 Collision* gazebo::physics::Contact::collision2

Pointer to the second collision object.

#### 10.27.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

#### 10.27.4.4 double gazebo::physics::Contact::depths[32]

Array of contact depths.

#### 10.27.4.5 math::Vector3 gazebo::physics::Contact::normals[32]

Array of force normals.

#### 10.27.4.6 math::Vector3 gazebo::physics::Contact::positions[32]

Array of force positions.

#### 10.27.4.7 common::Time gazebo::physics::Contact::time

Time at which the contact occurred.

#### 10.27.4.8 WorldPtr gazebo::physics::Contact::world

**World** (p. 1157) in which the contact occurred.

#### 10.27.4.9 JointWrench gazebo::physics::Contact::wrench[32]

Array of forces for the contact.

All forces and torques are relative to the center of mass of the respective links that the collision elments are attached to.

The documentation for this class was generated from the following file:

- **Contact.hh**

## 10.28 gazebo::physics::ContactManager Class Reference

Aggregates all the contact information generated by the collision detection engine.

`#include <physics/physics.hh>`

### Public Member Functions

- **ContactManager** ()
  *Constructor.*
- virtual ~**ContactManager** ()
  *Destructor.*
- void **Clear** ()
  *Clear all stored contacts.*
- std::string **CreateFilter** (const std::string &_topic, const std::vector< std::string > &_collisions)
  *Create a filter for contacts.*
- std::string **CreateFilter** (const std::string &_topic, const std::string &_collision)
  *Create a filter for contacts.*

- std::string **CreateFilter** (const std::string &_name, const std::map< std::string, **physics::CollisionPtr** > &_collisions)

  *Create a filter for contacts.*
- **Contact** * **GetContact** (unsigned int _index) const

  *Get a single contact by index.*
- unsigned int **GetContactCount** () const

  *Return the number of valid contacts.*
- const std::vector< **Contact** * > & **GetContacts** () const

  *Get all the contacts.*
- void **Init** (**WorldPtr** _world)

  *Initialize the **ContactManager** (p. 263).*
- **Contact** * **NewContact** (**Collision** *_collision1, **Collision** *_collision2, const **common::Time** &_time)

  *Add a new contact.*
- void **PublishContacts** ()

  *Publish all contacts in a msgs::Contacts message.*
- void **ResetCount** ()

  *Set the contact count to zero.*

### 10.28.1 Detailed Description

Aggregates all the contact information generated by the collision detection engine.

### 10.28.2 Constructor & Destructor Documentation

#### 10.28.2.1 gazebo::physics::ContactManager::ContactManager ( )

Constructor.

#### 10.28.2.2 virtual gazebo::physics::ContactManager::~ContactManager ( ) [virtual]

Destructor.

### 10.28.3 Member Function Documentation

#### 10.28.3.1 void gazebo::physics::ContactManager::Clear ( )

Clear all stored contacts.

#### 10.28.3.2 std::string gazebo::physics::ContactManager::CreateFilter ( const std::string & *_topic,* const std::vector< std::string > & *_collisions* )

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collisions. param[in] _name Filter name. param[in] _collisions A list of collision names used for filtering.

**Returns**

New topic where filtered messages will be published to.

#### 10.28.3.3 std::string gazebo::physics::ContactManager::CreateFilter ( const std::string & *_topic,* const std::string & *_collision* )

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collision. param[in] _name Filter name. param[in] _collision A collision name used for filtering.

**Returns**

New topic where filtered messages will be published to.

#### 10.28.3.4 std::string gazebo::physics::ContactManager::CreateFilter ( const std::string & *_name,* const std::map< std::string, physics::CollisionPtr > & *_collisions* )

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collision. param[in] _name Filter name. param[in] _collisions A map of collision name to collision object.

**Returns**

New topic where filtered messages will be published to.

#### 10.28.3.5 Contact* gazebo::physics::ContactManager::GetContact ( unsigned int *_index* ) const

Get a single contact by index.

The index must be between 0 and **ContactManager::GetContactCount** (p. 265).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the **Contact** (p. 260) to return. |

**Returns**

Pointer to a contact, NULL If index is invalid.

#### 10.28.3.6 unsigned int gazebo::physics::ContactManager::GetContactCount ( ) const

Return the number of valid contacts.

#### 10.28.3.7 const std::vector<Contact *>& gazebo::physics::ContactManager::GetContacts ( ) const

Get all the contacts.

The return vector may have invalid contacts. Only use contents of the vector between 0 and **ContactManager::GetContactCount** (p. 265)

**Returns**

Vector of contact pointers.

#### 10.28.3.8 void gazebo::physics::ContactManager::Init ( WorldPtr _world )

Initialize the **ContactManager** (p. 263).

This is required in order to publish contact messages via the **ContactManager::PublishContacts** (p. 266) method.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to the world that is initializing the contact manager. |

#### 10.28.3.9 Contact* gazebo::physics::ContactManager::NewContact ( Collision * _collision1, Collision * _collision2, const common::Time & _time )

Add a new contact.

Noramlly this is only used by a Physics/Collision engine when a new contact is generated. All other users should just make use of the accessor functions.

If no one is listening, then the return value will be NULL. This is a signal to the Physics engine that it can skip the extra processing necessary to get back contact information.

**Returns**

The new contact. The physics engine should populate the contact's parameters. NULL will be returned if there are no subscribers to the contact topic.

#### 10.28.3.10 void gazebo::physics::ContactManager::PublishContacts ( )

Publish all contacts in a msgs::Contacts message.

#### 10.28.3.11 void gazebo::physics::ContactManager::ResetCount ( )

Set the contact count to zero.

The documentation for this class was generated from the following file:

- **ContactManager.hh**

## 10.29 gazebo::physics::ContactPublisher Class Reference

A custom contact publisher created for each contact filter in the **Contact** (p. 260) Manager.

`#include <ContactManager.hh>`

### Public Attributes

- std::vector< std::string > **collisionNames**
- boost::unordered_set< **Collision** * > **collisions**

  *Pointers of collisions monitored by contact manager for contacts.*
- std::vector< **Contact** * > **contacts**

  *A list of contacts associated to the collisions.*

- **transport::PublisherPtr publisher**

  ***Contact*** *(p. 260) message publisher.*

### 10.29.1 Detailed Description

A custom contact publisher created for each contact filter in the **Contact** (p. 260) Manager.

### 10.29.2 Member Data Documentation

#### 10.29.2.1 std::vector<std::string> gazebo::physics::ContactPublisher::collisionNames

#### 10.29.2.2 boost::unordered_set<Collision *> gazebo::physics::ContactPublisher::collisions

Pointers of collisions monitored by contact manager for contacts.

#### 10.29.2.3 std::vector<Contact *> gazebo::physics::ContactPublisher::contacts

A list of contacts associated to the collisions.

#### 10.29.2.4 transport::PublisherPtr gazebo::physics::ContactPublisher::publisher

**Contact** (p. 260) message publisher.

The documentation for this class was generated from the following file:

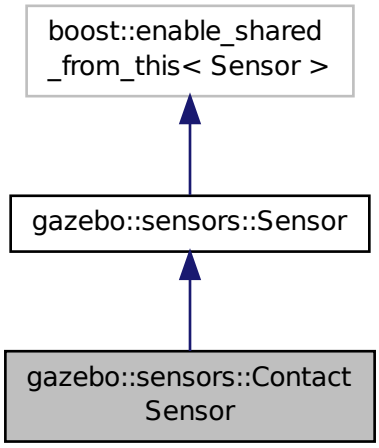
- **ContactManager.hh**

## 10.30 gazebo::sensors::ContactSensor Class Reference

Contact sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ContactSensor:




## Public Member Functions

- **ContactSensor** ()

  *Constructor.*
- virtual ~**ContactSensor** ()

  *Destructor.*
- unsigned int **GetCollisionContactCount** (const std::string &_collisionName) const

  *Return the number of contacts for an observed collision.*
- unsigned int **GetCollisionCount** () const

  *Get the number of collisions that the sensor is observing.*
- std::string **GetCollisionName** (unsigned int _index) const

  *Get a collision name at index _index.*
- msgs::Contacts **GetContacts** () const

  *Get all the contacts for the **ContactSensor*** (p. *267*).
- std::map< std::string,
  **physics::Contact** > **GetContacts** (const std::string &_collisionName)

  *Gets contacts of a collision.*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  *Load the sensor with SDF parameters.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

### Protected Member Functions

- virtual void **Fini** ()

  *Finalize the sensor.*
- virtual void **UpdateImpl** (bool _force)

  *Update the sensor information.*

### Additional Inherited Members

## 10.30.1 Detailed Description

Contact sensor.

This sensor detects and reports contacts between objects

## 10.30.2 Constructor & Destructor Documentation

#### 10.30.2.1 gazebo::sensors::ContactSensor::ContactSensor ( )

Constructor.

#### 10.30.2.2 virtual gazebo::sensors::ContactSensor::~ContactSensor ( ) [virtual]

Destructor.

## 10.30.3 Member Function Documentation

#### 10.30.3.1 virtual void gazebo::sensors::ContactSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.30.3.2 unsigned int gazebo::sensors::ContactSensor::GetCollisionContactCount ( const std::string & *_collisionName* ) const

Return the number of contacts for an observed collision.

**Parameters**

| | | |
|---|---|---|
| in | *_collisionName* | The name of the observed collision. |

**Returns**

The collision contact count.

#### 10.30.3.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount ( ) const

Get the number of collisions that the sensor is observing.

**Returns**

Number of collisions.

#### 10.30.3.4 std::string gazebo::sensors::ContactSensor::GetCollisionName ( unsigned int *_index* ) const

Get a collision name at index _index.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of collision in collection of collisions. |

**Returns**

name of collision.

#### 10.30.3.5 msgs::Contacts gazebo::sensors::ContactSensor::GetContacts ( ) const

Get all the contacts for the **ContactSensor** (p. 267).

**Returns**

Message that contains contact information between collision pairs.

During ODEPhysics::UpdateCollisions, all collision pairs in the world are pushed into a buffer within ContactManager. Subsequently, World::Update invokes ContactManager::PublishContacts to publish all contacts generated within a timestep onto Gazebo topic ~/physics/contacts.

Each **ContactSensor** (p. 267) subscribes to the Gazebo ~/physics/contacts topic, retrieves all contact pairs in a time step and filters them wthin ContactSensor::OnContacts against <collision> body name specified by the **ContactSensor** (p. 267) SDF. All collision pairs between **ContactSensor** (p. 267) <collision> body and other bodies in the world are stored in an array inside contacts.proto.

Within each element of the contact.proto array inside contacts.proto, list of collisions between collision bodies (collision1 and collision 2) are stored in an array of elements, (position, normal, depth, wrench). A timestamp has also been added (time). Details are described below:

- string collision1 name of the first collision object.
- string collision2 name of the second collision object.
- Vector3d position position of the contact joint in inertial frame.
- Vector3d normal normal of the contact joint in inertial frame.
- double depth intersection (penetration) depth of two collision bodies.
- JointWrench wrench Forces and torques acting on both collision bodies. See joint_wrench.proto for details. The forces and torques are applied at the CG of perspective links for each collision body, specified in the inertial frame.
- Time time time at which this contact happened.

#### 10.30.3.6 std::map<std::string, physics::Contact> gazebo::sensors::ContactSensor::GetContacts ( const std::string & _collisionName )

Gets contacts of a collision.

**Parameters**

| | | |
|---|---|---|
| in | *_collisionName* | Name of collision |

**Returns**

Container of contacts

#### 10.30.3.7 virtual void gazebo::sensors::ContactSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.30.3.8 virtual bool gazebo::sensors::ContactSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.30.3.9 virtual void gazebo::sensors::ContactSensor::Load ( const std::string & _worldName, sdf::ElementPtr _sdf ) [virtual]

Load the sensor with SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF **Sensor** (p. 837) parameters |
| in | *_worldName* | Name of world to load from |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.30.3.10 virtual void gazebo::sensors::ContactSensor::Load ( const std::string & _worldName ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.30.3.11 virtual void gazebo::sensors::ContactSensor::UpdateImpl ( bool *_force* ) [protected], [virtual]

Update the sensor information.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not. |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

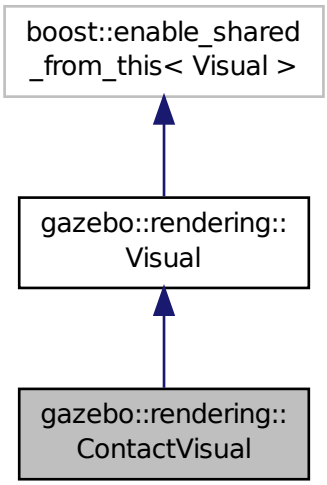
- **ContactSensor.hh**

## 10.31 gazebo::rendering::ContactVisual Class Reference

Contact visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ContactVisual:




### Public Member Functions

- **ContactVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
  *Constructor.*
- virtual ~**ContactVisual** ()
  *Destructor.*
- void **SetEnabled** (bool _enabled)
  *Set to true to enable contact visualization.*

### Additional Inherited Members

### 10.31.1 Detailed Description

Contact visualization.

This class visualizes contact points by drawing arrows in the 3D environment.

### 10.31.2 Constructor & Destructor Documentation

#### 10.31.2.1 gazebo::rendering::ContactVisual::ContactVisual ( const std::string & *_name*, VisualPtr *_vis*, const std::string & *_topicName* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the **ContactVisual** (p. 272) |
| in | *_vis* | Pointer the parent **Visual** (p. 1121) |
| in | *_topicName* | Name of the topic which publishes the contact information. |

#### 10.31.2.2 virtual gazebo::rendering::ContactVisual::~ContactVisual ( ) [virtual]

Destructor.

### 10.31.3 Member Function Documentation

#### 10.31.3.1 void gazebo::rendering::ContactVisual::SetEnabled ( bool *_enabled* )

Set to true to enable contact visualization.

**Parameters**

| | | |
|---|---|---|
| in | *_enabled* | True to show contacts, false to hide. |

The documentation for this class was generated from the following file:

- **ContactVisual.hh**

## 10.32 gazebo::rendering::Conversions Class Reference

**Conversions** (p. 273) **Conversions.hh** (p. 1218) **rendering/Conversions.hh** (p. 1218).

```
#include <Conversions.hh>
```

### Static Public Member Functions

- static Ogre::ColourValue **Convert** (const **common::Color** &_clr)

  *Return the equivalent ogre color.*

- static **common::Color Convert** (const Ogre::ColourValue &_clr)

  *Return the equivalent gazebo color.*
- static Ogre::Vector3 **Convert** (const **math::Vector3** &_v)

  *return **Ogre** (p. 130) Vector from Gazebo Vector3*
- static **math::Vector3 Convert** (const Ogre::Vector3 &_v)

  *return gazebo Vector from ogre Vector3*
- static Ogre::Quaternion **Convert** (const **math::Quaternion** &_v)

  *Gazebo quaternion to **Ogre** (p. 130) quaternion.*
- static **math::Quaternion Convert** (const Ogre::Quaternion &_v)

  ***Ogre** (p. 130) quaternion to Gazebo quaternion.*

### 10.32.1 Detailed Description

**Conversions** (p. 273) **Conversions.hh** (p. 1218) **rendering/Conversions.hh** (p. 1218).

A set of utility function to convert between Gazebo and **Ogre** (p. 130) data types

### 10.32.2 Member Function Documentation

#### 10.32.2.1 static Ogre::ColourValue gazebo::rendering::Conversions::Convert ( const common::Color & _clr ) [static]

Return the equivalent ogre color.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | Gazebo color to convert |

**Returns**

**Ogre** (p. 130) color value

#### 10.32.2.2 static common::Color gazebo::rendering::Conversions::Convert ( const Ogre::ColourValue & _clr ) [static]

Return the equivalent gazebo color.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | **Ogre** (p. 130) color to convert |

**Returns**

Gazebo color value

#### 10.32.2.3 static Ogre::Vector3 gazebo::rendering::Conversions::Convert ( const math::Vector3 & _v ) [static]

return **Ogre** (p. 130) Vector from Gazebo Vector3

**Parameters**

| | | |
|---|---|---|
| in | _v | Gazebo vector |

**Returns**

**Ogre** (p. 130) vector

#### 10.32.2.4 static math::Vector3 gazebo::rendering::Conversions::Convert ( const Ogre::Vector3 & _v ) [static]

return gazebo Vector from ogre Vector3

**Parameters**

| | | |
|---|---|---|
| in | _v | **Ogre** (p. 130) vector |

**Returns**

Gazebo vector

#### 10.32.2.5 static Ogre::Quaternion gazebo::rendering::Conversions::Convert ( const math::Quaternion & _v ) [static]

Gazebo quaternion to **Ogre** (p. 130) quaternion.

**Parameters**

| | | |
|---|---|---|
| in | _v | Gazebo quaternion |

**Returns**

**Ogre** (p. 130) quaternion

#### 10.32.2.6 static math::Quaternion gazebo::rendering::Conversions::Convert ( const Ogre::Quaternion & _v ) [static]

**Ogre** (p. 130) quaternion to Gazebo quaternion.

**Parameters**

| | | |
|---|---|---|
| in | _v | **Ogre** (p. 130) quaternion return Gazebo quaternion |

The documentation for this class was generated from the following file:

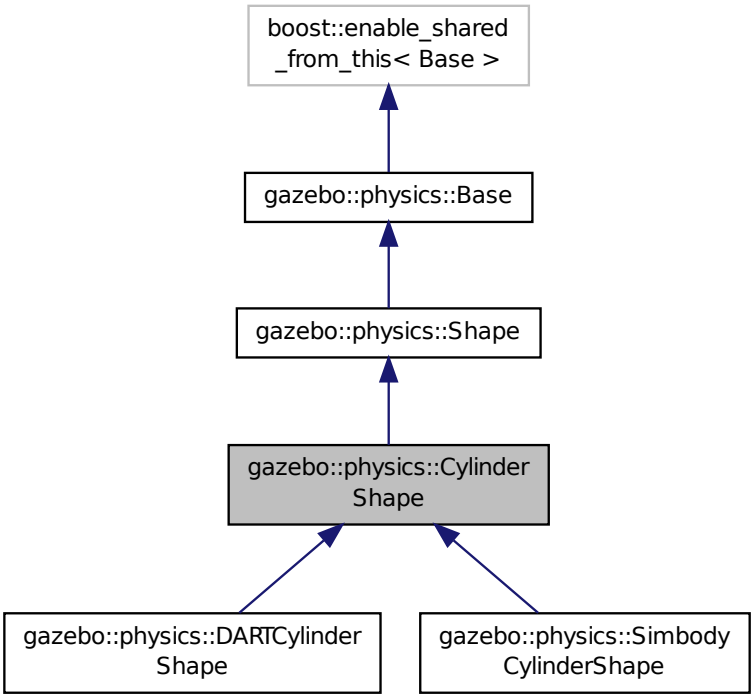
- **Conversions.hh**

## 10.33 gazebo::physics::CylinderShape Class Reference

Cylinder collision.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::CylinderShape:




## Public Member Functions

- **CylinderShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**CylinderShape** ()

  *Destructor.*
- void **FillMsg** (msgs::Geometry &_msg)

  *Fill in the values for a geomertry message.*
- double **GetLength** () const

  *Get length.*
- double **GetRadius** () const

  *Get radius.*
- void **Init** ()

  *Initialize the cylinder.*
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *Update values based on a message.*
- void **SetLength** (double _length)

  *Set length.*
- void **SetRadius** (double _radius)

*Set radius.*

- virtual void **SetScale** (const **math::Vector3** &_scale)

  *Set scale of cylinder.*

- virtual void **SetSize** (double _radius, double _length)

  *Set the size of the cylinder.*

## Additional Inherited Members

### 10.33.1 Detailed Description

Cylinder collision.

### 10.33.2 Constructor & Destructor Documentation

#### 10.33.2.1 gazebo::physics::CylinderShape::CylinderShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the shape. |

#### 10.33.2.2 virtual gazebo::physics::CylinderShape::~CylinderShape ( ) [virtual]

Destructor.

### 10.33.3 Member Function Documentation

#### 10.33.3.1 void gazebo::physics::CylinderShape::FillMsg ( msgs::Geometry & *_msg* ) [virtual]

Fill in the values for a geomertry message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | The geometry message to fill. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.33.3.2 double gazebo::physics::CylinderShape::GetLength ( ) const

Get length.

**Returns**

The cylinder length.

#### 10.33.3.3 double gazebo::physics::CylinderShape::GetRadius ( ) const

Get radius.

**Returns**

The cylinder radius.

#### 10.33.3.4 void gazebo::physics::CylinderShape::Init ( ) [virtual]

Initialize the cylinder.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.33.3.5 virtual void gazebo::physics::CylinderShape::ProcessMsg ( const msgs::Geometry & *_msg* ) [virtual]

Update values based on a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to update from. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.33.3.6 void gazebo::physics::CylinderShape::SetLength ( double *_length* )

Set length.

**Parameters**

| | | |
|---|---|---|
| in | *_length* | New length of the cylinder. |

#### 10.33.3.7 void gazebo::physics::CylinderShape::SetRadius ( double *_radius* )

Set radius.

**Parameters**

| | |
|---|---|
| *in}* | _radius New radius of the cylinder. |

#### 10.33.3.8 virtual void gazebo::physics::CylinderShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set scale of cylinder.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the cylinder to. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.33.3.9 virtual void gazebo::physics::CylinderShape::SetSize ( double *_radius,* double *_length* ) [virtual]

Set the size of the cylinder.

**Parameters**

| | | |
|---|---|---|
| in | *_radius* | New radius. |
| in | *_lenght* | New length. |

Reimplemented in **gazebo::physics::SimbodyCylinderShape** (p. 877), and **gazebo::physics::DARTCylinderShape** (p. 292).

Referenced by gazebo::physics::DARTCylinderShape::SetSize(), and gazebo::physics::SimbodyCylinderShape::SetSize().

The documentation for this class was generated from the following file:

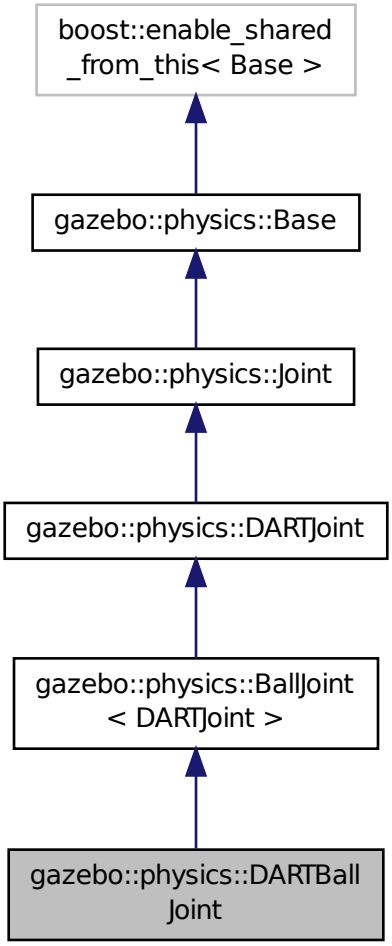
- **CylinderShape.hh**

## 10.34 gazebo::physics::DARTBallJoint Class Reference

An **DARTBallJoint** (p. 279).

`#include <DARTBallJoint.hh>`

Inheritance diagram for gazebo::physics::DARTBallJoint:




## Public Member Functions

- **DARTBallJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**DARTBallJoint** ()

  *Destructor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual **math::Angle GetAngleImpl** (int) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int) const

  *Get the axis of rotation in global cooridnate frame.*

- virtual double **GetMaxForce** (int)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint*** (p. *496*) *from a SDF sdf::Element.*
- virtual void **SetMaxForce** (int, double)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int, double)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- void **SetForceImpl** (int, double)

  *Set the force applied to this **physics::Joint*** (p. *496*).

### Protected Attributes

- dart::dynamics::BallJoint ∗ **dtBallJoint**

### Additional Inherited Members

## 10.34.1 Detailed Description

An **DARTBallJoint** (p. 279).

## 10.34.2 Constructor & Destructor Documentation

### 10.34.2.1 gazebo::physics::DARTBallJoint::DARTBallJoint ( BasePtr *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the **Joint** (p. 496) |

### 10.34.2.2 virtual gazebo::physics::DARTBallJoint::∼DARTBallJoint ( ) [virtual]

Destructor.

## 10.34.3 Member Function Documentation

#### 10.34.3.1 virtual math::Vector3 gazebo::physics::DARTBallJoint::GetAnchor ( int _index ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 502).

#### 10.34.3.2 virtual math::Angle gazebo::physics::DARTBallJoint::GetAngleImpl ( int _index ) const [inline], [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.34.3.3 virtual math::Vector3 gazebo::physics::DARTBallJoint::GetGlobalAxis ( int _index ) const [inline], [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.34.3.4 virtual double gazebo::physics::DARTBallJoint::GetMaxForce ( int _index ) [inline], [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.34.3.5 virtual double gazebo::physics::DARTBallJoint::GetVelocity ( int *_index* ) const [inline], [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.34.3.6 virtual void gazebo::physics::DARTBallJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 311).

#### 10.34.3.7 virtual void gazebo::physics::DARTBallJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load **physics::Joint** (p. 496) from a SDF sdf::Element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::DARTJoint** (p. 311).

#### 10.34.3.8 void gazebo::physics::DARTBallJoint::SetForceImpl ( int *_index*, double *_force* ) [inline], [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. |

Implements **gazebo::physics::DARTJoint** (p. 312).

References gzerr.

#### 10.34.3.9 virtual void gazebo::physics::DARTBallJoint::SetMaxForce ( int *_index*, double *_force* ) [inline], [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.34.3.10 virtual void gazebo::physics::DARTBallJoint::SetVelocity ( int *_index*, double *_vel* ) [inline], [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

### 10.34.4 Member Data Documentation

#### 10.34.4.1 dart::dynamics::BallJoint* gazebo::physics::DARTBallJoint::dtBallJoint [protected]

The documentation for this class was generated from the following file:

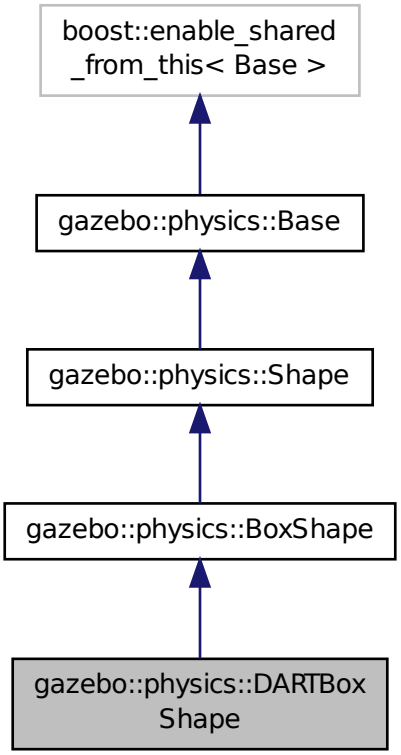
- **DARTBallJoint.hh**

## 10.35 gazebo::physics::DARTBoxShape Class Reference

DART Box shape.

`#include <DARTBoxShape.hh>`

Inheritance diagram for gazebo::physics::DARTBoxShape:




### Public Member Functions

- **DARTBoxShape** (**DARTCollisionPtr** _parent)

  *Constructor.*
- virtual ~**DARTBoxShape** ()

  *Destructor.*
- virtual void **SetSize** (const **math::Vector3** &_size)

  *Set the size of the box.*

### Additional Inherited Members

#### 10.35.1 Detailed Description

DART Box shape.

#### 10.35.2 Constructor & Destructor Documentation

#### 10.35.2.1 gazebo::physics::DARTBoxShape::DARTBoxShape ( DARTCollisionPtr _parent ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent **Collision** (p. 220). |

#### 10.35.2.2 virtual gazebo::physics::DARTBoxShape::~DARTBoxShape ( ) [inline], [virtual]

Destructor.

### 10.35.3 Member Function Documentation

#### 10.35.3.1 virtual void gazebo::physics::DARTBoxShape::SetSize ( const math::Vector3 & _size ) [inline], [virtual]

Set the size of the box.

**Parameters**

| | | |
|---|---|---|
| in | *_size* | Size of each side of the box. |

Reimplemented from **gazebo::physics::BoxShape** (p. 179).

References gazebo::physics::Shape::collisionParent, gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::equal(), gazebo::physics::DARTCollision::GetDARTBodyNode(), gzerr, gzwarn, NULL, gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

The documentation for this class was generated from the following file:

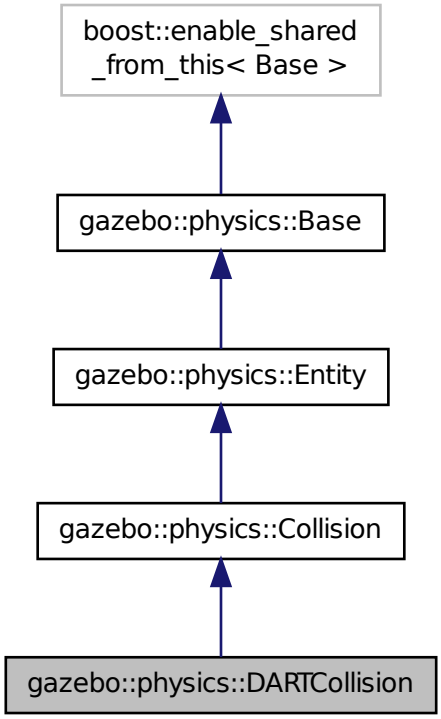
- **DARTBoxShape.hh**

## 10.36 gazebo::physics::DARTCollision Class Reference

**Base** (p. 159) class for all DART collisions.

`#include <DARTCollision.hh>`

Inheritance diagram for gazebo::physics::DARTCollision:




## Public Member Functions

- **DARTCollision** (**LinkPtr** _parent)

  *Constructor.*
- virtual ~**DARTCollision** ()

  *Destructor.*
- virtual void **Fini** ()

  *Finalize the collision.*
- virtual **math::Box GetBoundingBox** () const

  *Get the bounding box for this collision.*
- virtual unsigned int **GetCategoryBits** () const

  *Get the category bits, used during collision detection.*
- virtual unsigned int **GetCollideBits** () const

  *Get the collide bits, used during collision detection.*
- dart::dynamics::BodyNode * **GetDARTBodyNode** () const

  *Get DART body node.*
- dart::dynamics::Shape * **GetDARTCollisionShape** () const

  *Get DART collision shape.*

- virtual void **Init** ()

  *Initialize the collision.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the collision.*
- virtual void **OnPoseChange** ()

  *This function is called when the entity's (or one of its parents) pose of the parent has changed.*
- virtual void **SetCategoryBits** (unsigned int _bits)

  *Set the category bits, used during collision detection.*
- virtual void **SetCollideBits** (unsigned int _bits)

  *Set the collide bits, used during collision detection.*
- void **SetDARTCollisionShape** (dart::dynamics::Shape *_shape, bool _placeable=true)

  *Set DART collision shape.*

### Additional Inherited Members

### 10.36.1 Detailed Description

**Base** (p. 159) class for all DART collisions.

### 10.36.2 Constructor & Destructor Documentation

#### 10.36.2.1 gazebo::physics::DARTCollision::DARTCollision ( LinkPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_link* | Parent **Link** (p. 542) |

#### 10.36.2.2 virtual gazebo::physics::DARTCollision::~DARTCollision ( ) [virtual]

Destructor.

### 10.36.3 Member Function Documentation

#### 10.36.3.1 virtual void gazebo::physics::DARTCollision::Fini ( ) [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Collision** (p. 223).

#### 10.36.3.2 virtual math::Box gazebo::physics::DARTCollision::GetBoundingBox ( ) const [virtual]

Get the bounding box for this collision.

**Returns**

The bounding box.

Implements **gazebo::physics::Collision** (p. 223).

#### 10.36.3.3 virtual unsigned int gazebo::physics::DARTCollision::GetCategoryBits ( ) const [virtual]

Get the category bits, used during collision detection.

**Returns**

The bits

#### 10.36.3.4 virtual unsigned int gazebo::physics::DARTCollision::GetCollideBits ( ) const [virtual]

Get the collide bits, used during collision detection.

**Returns**

The bits

#### 10.36.3.5 dart::dynamics::BodyNode* gazebo::physics::DARTCollision::GetDARTBodyNode ( ) const

Get DART body node.

**Returns**

Pointer to the dart BodyNode.

Referenced by gazebo::physics::DARTPlaneShape::CreatePlane(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::DARTCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

#### 10.36.3.6 dart::dynamics::Shape* gazebo::physics::DARTCollision::GetDARTCollisionShape ( ) const

Get DART collision shape.

#### 10.36.3.7 virtual void gazebo::physics::DARTCollision::Init ( ) [virtual]

Initialize the collision.

Reimplemented from **gazebo::physics::Collision** (p. 226).

#### 10.36.3.8 virtual void gazebo::physics::DARTCollision::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF to load from. |

Reimplemented from **gazebo::physics::Collision** (p. 227).

#### 10.36.3.9 virtual void gazebo::physics::DARTCollision::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **gazebo::physics::Entity** (p. 386).

#### 10.36.3.10 virtual void gazebo::physics::DARTCollision::SetCategoryBits ( unsigned int *_bits* ) [virtual]

Set the category bits, used during collision detection.

**Parameters**

| | | |
|---|---|---|
| in | *_bits* | The bits to set. |

Implements **gazebo::physics::Collision** (p. 227).

#### 10.36.3.11 virtual void gazebo::physics::DARTCollision::SetCollideBits ( unsigned int *_bits* ) [virtual]

Set the collide bits, used during collision detection.

**Parameters**

| | | |
|---|---|---|
| in | *_bits* | The bits to set. |

Implements **gazebo::physics::Collision** (p. 227).

#### 10.36.3.12 void gazebo::physics::DARTCollision::SetDARTCollisionShape ( dart::dynamics::Shape * *_shape,* bool *_placeable* = true )

Set DART collision shape.

**Parameters**

| | | |
|---|---|---|
| in | *_shape* | DART **Collision** (p. 220) shape |
| in | *_placeable* | True to make the object movable. |

The documentation for this class was generated from the following file:

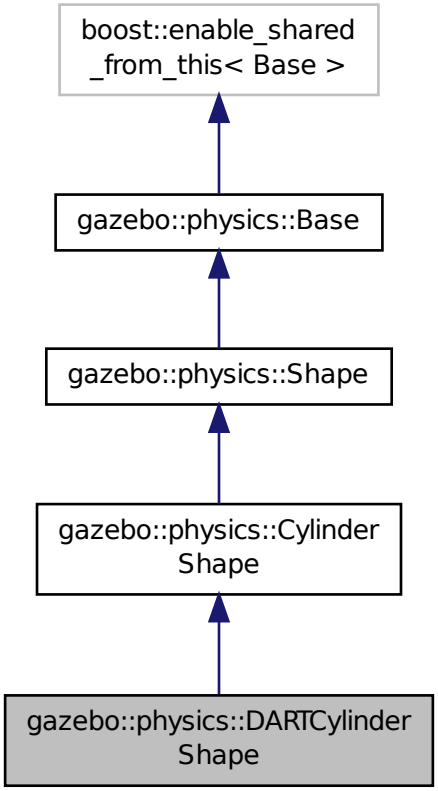
- **DARTCollision.hh**

## 10.37 gazebo::physics::DARTCylinderShape Class Reference

DART cylinder shape.

`#include <DARTCylinderShape.hh>`

Inheritance diagram for gazebo::physics::DARTCylinderShape:




## Public Member Functions

- **DARTCylinderShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**DARTCylinderShape** ()

  *Destructor.*
- void **SetSize** (double _radius, double _length)

  *Set the size of the cylinder.*

## Additional Inherited Members

### 10.37.1 Detailed Description

DART cylinder shape.

### 10.37.2 Constructor & Destructor Documentation

#### 10.37.2.1 gazebo::physics::DARTCylinderShape::DARTCylinderShape ( CollisionPtr *_parent* ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | **Collision** (p. 220) parent. |

#### 10.37.2.2 virtual gazebo::physics::DARTCylinderShape::~DARTCylinderShape ( ) [inline], [virtual]

Destructor.

### 10.37.3 Member Function Documentation

#### 10.37.3.1 void gazebo::physics::DARTCylinderShape::SetSize ( double *_radius,* double *_length* ) [inline], [virtual]

Set the size of the cylinder.

**Parameters**

| | | |
|---|---|---|
| in | *_radius* | New radius. |
| in | *_lenght* | New length. |

Reimplemented from **gazebo::physics::CylinderShape** (p. 279).

References gazebo::physics::Shape::collisionParent, gazebo::math::equal(), gazebo::physics::DARTCollision::GetDARTBodyNode(), gzerr, gzwarn, NULL, and gazebo::physics::CylinderShape::SetSize().

The documentation for this class was generated from the following file:

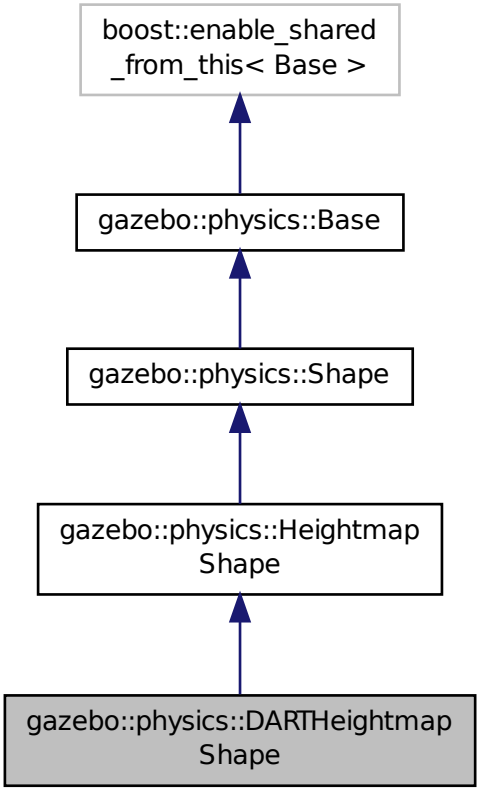
- **DARTCylinderShape.hh**

## 10.38 gazebo::physics::DARTHeightmapShape Class Reference

DART Height map collision.

`#include <DARTHeightmapShape.hh>`

Inheritance diagram for gazebo::physics::DARTHeightmapShape:




## Public Member Functions

- **DARTHeightmapShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**DARTHeightmapShape** ()

  *Destructor.*
- virtual void **Init** ()

  *Initialize the heightmap.*

## Additional Inherited Members

### 10.38.1 Detailed Description

DART Height map collision.

### 10.38.2 Constructor & Destructor Documentation

#### 10.38.2.1 gazebo::physics::DARTHeightmapShape::DARTHeightmapShape ( CollisionPtr _parent )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | **Collision** (p. 220) parent. |

#### 10.38.2.2 virtual gazebo::physics::DARTHeightmapShape::~DARTHeightmapShape ( ) [virtual]

Destructor.

### 10.38.3 Member Function Documentation

#### 10.38.3.1 virtual void gazebo::physics::DARTHeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 469).

The documentation for this class was generated from the following file:

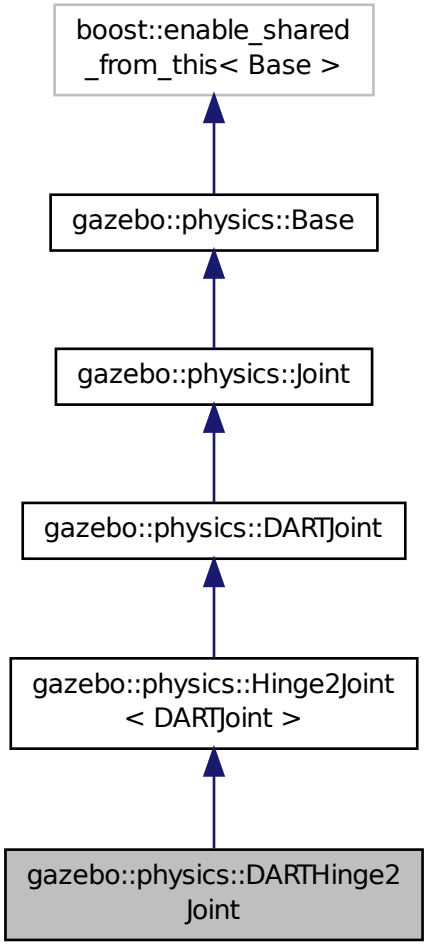
- **DARTHeightmapShape.hh**

## 10.39 gazebo::physics::DARTHinge2Joint Class Reference

A two axis hinge joint.

`#include <DARTHinge2Joint.hh>`

Inheritance diagram for gazebo::physics::DARTHinge2Joint:




## Public Member Functions

- **DARTHinge2Joint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**DARTHinge2Joint** ()

  *Destructor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*

- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the joint.*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetMaxForce** (int _index, double _force)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _vel)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 496).*

### Protected Attributes

- dart::dynamics::UniversalJoint * **dtUniveralJoint**

  *Universal joint of DART.*

### Additional Inherited Members

## 10.39.1 Detailed Description

A two axis hinge joint.

## 10.39.2 Constructor & Destructor Documentation

#### 10.39.2.1 gazebo::physics::DARTHinge2Joint::DARTHinge2Joint ( **BasePtr** *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the **Joint** (p. 496) |

#### 10.39.2.2 virtual gazebo::physics::DARTHinge2Joint::~DARTHinge2Joint ( ) [virtual]

Destructor.

### 10.39.3 Member Function Documentation

#### 10.39.3.1 virtual math::Vector3 gazebo::physics::DARTHinge2Joint::GetAnchor ( int _index ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 502).

#### 10.39.3.2 virtual math::Angle gazebo::physics::DARTHinge2Joint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.39.3.3 virtual math::Vector3 gazebo::physics::DARTHinge2Joint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.39.3.4 virtual double gazebo::physics::DARTHinge2Joint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.39.3.5 virtual double gazebo::physics::DARTHinge2Joint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.39.3.6 virtual void gazebo::physics::DARTHinge2Joint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 311).

#### 10.39.3.7 virtual void gazebo::physics::DARTHinge2Joint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the joint.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::Hinge2Joint**< **DARTJoint** > (p. 472).

#### 10.39.3.8 virtual void gazebo::physics::DARTHinge2Joint::SetAxis ( int *_index*, const **math::Vector3** & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.39.3.9 virtual void gazebo::physics::DARTHinge2Joint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. |

Implements **gazebo::physics::DARTJoint** (p. 312).

#### 10.39.3.10 virtual void gazebo::physics::DARTHinge2Joint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.39.3.11 virtual void gazebo::physics::DARTHinge2Joint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

### 10.39.4 Member Data Documentation

#### 10.39.4.1 dart::dynamics::UniversalJoint* gazebo::physics::DARTHinge2Joint::dtUniveralJoint [protected]

Universal joint of DART.

The documentation for this class was generated from the following file:

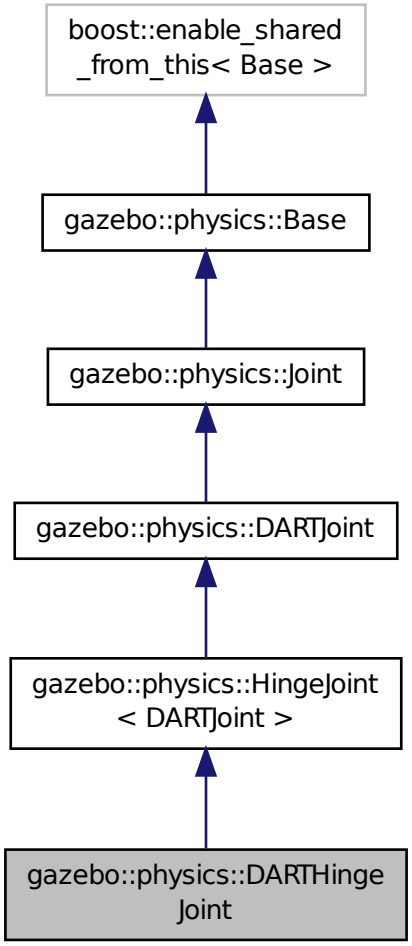
- **DARTHinge2Joint.hh**

## 10.40 gazebo::physics::DARTHingeJoint Class Reference

A single axis hinge joint.

`#include <DARTHingeJoint.hh>`

Inheritance diagram for gazebo::physics::DARTHingeJoint:




## Public Member Functions

- **DARTHingeJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**DARTHingeJoint** ()

  *Destructor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*

- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load joint.*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetMaxForce** (int _index, double _force)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _vel)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 496).*

### Protected Attributes

- dart::dynamics::RevoluteJoint * **dtRevoluteJoint**

  *Revolute joint of DART.*

### Additional Inherited Members

## 10.40.1 Detailed Description

A single axis hinge joint.

## 10.40.2 Constructor & Destructor Documentation

#### 10.40.2.1 gazebo::physics::DARTHingeJoint::DARTHingeJoint ( BasePtr *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the **Joint** (p. 496) |

#### 10.40.2.2 virtual gazebo::physics::DARTHingeJoint::~DARTHingeJoint ( ) [virtual]

Destructor.

### 10.40.3 Member Function Documentation

#### 10.40.3.1 virtual math::Vector3 gazebo::physics::DARTHingeJoint::GetAnchor ( int _index ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 502).

#### 10.40.3.2 virtual math::Angle gazebo::physics::DARTHingeJoint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.40.3.3 virtual math::Vector3 gazebo::physics::DARTHingeJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.40.3.4 virtual double gazebo::physics::DARTHingeJoint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.40.3.5 virtual double gazebo::physics::DARTHingeJoint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.40.3.6 virtual void gazebo::physics::DARTHingeJoint::Init ( ) [virtual]

Initialize joint.

Reimplemented from **gazebo::physics::HingeJoint< DARTJoint >** (p. 474).

#### 10.40.3.7 virtual void gazebo::physics::DARTHingeJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load joint.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to SDF element |

Reimplemented from **gazebo::physics::HingeJoint< DARTJoint >** (p. 474).

#### 10.40.3.8 virtual void gazebo::physics::DARTHingeJoint::SetAxis ( int *_index*, const math::Vector3 & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.40.3.9 virtual void gazebo::physics::DARTHingeJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. |

Implements **gazebo::physics::DARTJoint** (p. 312).

#### 10.40.3.10 virtual void gazebo::physics::DARTHingeJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.40.3.11 virtual void gazebo::physics::DARTHingeJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

### 10.40.4 Member Data Documentation

#### 10.40.4.1 dart::dynamics::RevoluteJoint* gazebo::physics::DARTHingeJoint::dtRevoluteJoint [protected]

Revolute joint of DART.

The documentation for this class was generated from the following file:

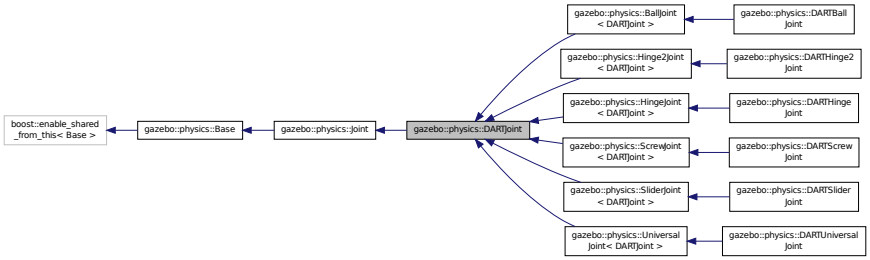
- **DARTHingeJoint.hh**

## 10.41 gazebo::physics::DARTJoint Class Reference

DART joint interface.

`#include <DARTJoint.hh>`

Inheritance diagram for gazebo::physics::DARTJoint:




## Public Member Functions

- **DARTJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ∼**DARTJoint** ()

  *Destructor.*
- virtual void **ApplyDamping** ()

  *Callback to apply damping force to joint.*
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const

  *Determines of the two bodies are connected by a joint.*
- virtual void **Attach** (**LinkPtr** _parent, **LinkPtr** _child)

  *Attach the two bodies with this joint.*
- virtual void **Detach** ()

  *Detach this joint from all links.*
- virtual unsigned int **GetAngleCount** () const

  *Get the angle count.*
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index)

  *Get a non-generic parameter for the joint.*
- dart::dynamics::Joint ∗ **GetDARTJoint** ()

  *Get DART joint pointer.*
- **DARTModelPtr GetDARTModel** () const

  *Get DART model pointer.*
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)

  *get internal force and torque values at a joint.*
- virtual **math::Angle GetHighStop** (int _index)

  *Get the high stop of an axis(index).*
- virtual **LinkPtr GetJointLink** (int _index) const

  *Get the link to which the joint is attached according the _index.*
- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const

  *Get the forces applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const

  *Get the torque applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).*

- virtual **math::Angle GetLowStop** (int _index)

  *Get the low stop of an axis(index).*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint*** (p. *496*) *from a SDF sdf::Element.*
- virtual void **Reset** ()

  *Reset the joint.*
- virtual void **SetAnchor** (int, const **gazebo::math::Vector3** &)

  *Set the anchor point.*
- virtual void **SetAttribute** (const std::string &_key, int _index, const boost::any &_value)

  *Set a non-generic parameter for the joint.*
- virtual void **SetDamping** (int _index, double _damping)

  *Set the joint damping.*
- virtual void **SetForce** (int _index, double _force)

  *Set the force applied to this **physics::Joint*** (p. *496*).
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*

### Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _force)=0

  *Set the force applied to this **physics::Joint*** (p. *496*).

### Protected Attributes

- **DARTPhysicsPtr dartPhysicsEngine**

  ***DARTPhysics*** (p. *330*) *engine pointer.*
- dart::dynamics::BodyNode * **dtChildBodyNode**

  *DART child body node pointer.*
- dart::dynamics::Joint * **dtJoint**

  *DART joint pointer.*

### Additional Inherited Members

### 10.41.1 Detailed Description

DART joint interface.

## 10.41.2 Constructor & Destructor Documentation

### 10.41.2.1 gazebo::physics::DARTJoint::DARTJoint ( BasePtr _parent )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the **Joint** (p. 496). |

### 10.41.2.2 virtual gazebo::physics::DARTJoint::~DARTJoint ( ) [virtual]

Destructor.

## 10.41.3 Member Function Documentation

### 10.41.3.1 virtual void gazebo::physics::DARTJoint::ApplyDamping ( ) [virtual]

Callback to apply damping force to joint.

Reimplemented from **gazebo::physics::Joint** (p. 501).

### 10.41.3.2 virtual bool gazebo::physics::DARTJoint::AreConnected ( LinkPtr _one, LinkPtr _two ) const [virtual]

Determines of the two bodies are connected by a joint.

**Parameters**

| | | |
|---|---|---|
| in | *_one* | First link. |
| in | *_two* | Second link. |

**Returns**

True if the two links are connected by a joint.

Implements **gazebo::physics::Joint** (p. 501).

### 10.41.3.3 virtual void gazebo::physics::DARTJoint::Attach ( LinkPtr _parent, LinkPtr _child ) [virtual]

Attach the two bodies with this joint.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent link. |
| in | *_child* | Child link. |

Reimplemented from **gazebo::physics::Joint** (p. 501).

#### 10.41.3.4 virtual void gazebo::physics::DARTJoint::Detach ( ) [virtual]

Detach this joint from all links.

Reimplemented from **gazebo::physics::Joint** (p. 502).

#### 10.41.3.5 virtual unsigned int gazebo::physics::DARTJoint::GetAngleCount ( ) const [virtual]

Get the angle count.

**Returns**

The number of DOF for the joint.

Implements **gazebo::physics::Joint** (p. 503).

Reimplemented in **gazebo::physics::BallJoint< DARTJoint >** (p. 158), **gazebo::physics::SliderJoint< DARTJoint >** (p. 975), **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 472), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1065), and **gazebo::physics::HingeJoint< DARTJoint >** (p. 474).

#### 10.41.3.6 virtual double gazebo::physics::DARTJoint::GetAttribute ( const std::string & *_key*, unsigned int *_index* ) [virtual]

Get a non-generic parameter for the joint.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | String key. |
| in | *_index* | Index of the axis. |
| in | *_value* | Value of the attribute. |

Implements **gazebo::physics::Joint** (p. 504).

#### 10.41.3.7 dart::dynamics::Joint* gazebo::physics::DARTJoint::GetDARTJoint ( )

Get DART joint pointer.

**Returns**

A pointer to the DART joint.

#### 10.41.3.8 DARTModelPtr gazebo::physics::DARTJoint::GetDARTModel ( ) const

Get DART model pointer.

**Returns**

A pointer to the DART model.

#### 10.41.3.9 virtual double gazebo::physics::DARTJoint::GetForce ( unsigned int *_index* ) [virtual]

**Todo** : not yet implemented. Get external forces applied at this **Joint** (p. 496). Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The force applied to an axis.

Reimplemented from **gazebo::physics::Joint** (p. 505).

#### 10.41.3.10 virtual JointWrench gazebo::physics::DARTJoint::GetForceTorque ( int *_index* ) [virtual]

#### 10.41.3.11 virtual JointWrench gazebo::physics::DARTJoint::GetForceTorque ( unsigned int *_index* ) [virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 529) data structure. Where **JointWrench.body1Force** (p. 530) contains the force applied by the parent **Link** (p. 542) on the **Joint** (p. 496) specified in the parent **Link** (p. 542) frame, and **JointWrench.body2Force** (p. 531) contains the force applied by the child **Link** (p. 542) on the **Joint** (p. 496) specified in the child **Link** (p. 542) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 496) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 542) frames.

Note that for ODE you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Not used right now |

**Returns**

The force and torque at the joint, see above for details on conventions.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.41.3.12 virtual math::Angle gazebo::physics::DARTJoint::GetHighStop ( int *_index* ) [virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 506).

Reimplemented in **gazebo::physics::BallJoint< DARTJoint >** (p. 158).

#### 10.41.3.13 virtual LinkPtr gazebo::physics::DARTJoint::GetJointLink ( int _index ) const [virtual]

Get the link to which the joint is attached according the _index.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the link to retreive. |

**Returns**

Pointer to the request link. NULL if the index was invalid.

Implements **gazebo::physics::Joint** (p. 507).

#### 10.41.3.14 virtual math::Vector3 gazebo::physics::DARTJoint::GetLinkForce ( unsigned int _index ) const [virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of the link(0 or 1). |

**Returns**

Force applied to the link.

Implements **gazebo::physics::Joint** (p. 507).

#### 10.41.3.15 virtual math::Vector3 gazebo::physics::DARTJoint::GetLinkTorque ( unsigned int _index ) const [virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).

Note that the unit of torque should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of the link(0 or 1) |

**Returns**

Torque applied to the link.

Implements **gazebo::physics::Joint** (p. 507).

#### 10.41.3.16 virtual math::Angle gazebo::physics::DARTJoint::GetLowStop ( int _index ) [virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 508).

Reimplemented in **gazebo::physics::BallJoint< DARTJoint >** (p. 158).

#### 10.41.3.17 virtual void gazebo::physics::DARTJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 510).

Reimplemented in **gazebo::physics::HingeJoint< DARTJoint >** (p. 474), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::DARTBallJoint** (p. 283), **gazebo::physics::DARTScrewJoint** (p. 343), **gazebo::physics::DARTSliderJoint** (p. 348), and **gazebo::physics::DARTUniversalJoint** (p. 356).

#### 10.41.3.18 virtual void gazebo::physics::DARTJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load **physics::Joint** (p. 496) from a SDF sdf::Element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::Joint** (p. 510).

Reimplemented in **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 472), **gazebo::physics::BallJoint< DARTJoint >** (p. 158), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1065), **gazebo::physics::HingeJoint< DARTJoint >** (p. 474), **gazebo::physics::SliderJoint< DARTJoint >** (p. 975), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::DARTBallJoint** (p. 283), **gazebo::physics::DARTScrewJoint** (p. 343), **gazebo::physics::DARTSliderJoint** (p. 348), and **gazebo::physics::DARTUniversalJoint** (p. 356).

#### 10.41.3.19 virtual void gazebo::physics::DARTJoint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 511).

#### 10.41.3.20 virtual void gazebo::physics::DARTJoint::SetAnchor ( int , const gazebo::math::Vector3 & ) [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 511).

Reimplemented in **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), and **gazebo::physics::SliderJoint< DARTJoint >** (p. 975).

#### 10.41.3.21 virtual void gazebo::physics::DARTJoint::SetAttribute ( const std::string & *_key*, int *_index*, const boost::any & *_value* ) [virtual]

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

**Parameters**

| | | |
|---|---|---|
| in | *_key* | String key. |
| in | *_index* | Index of the axis. |
| in | *_value* | Value of the attribute. |

Implements **gazebo::physics::Joint** (p. 511).

#### 10.41.3.22 virtual void gazebo::physics::DARTJoint::SetDamping ( int *_index*, double *_damping* ) [virtual]

Set the joint damping.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set, currently ignored, to be implemented. |
| in | *_damping* | Damping value for the axis. |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.41.3.23 virtual void gazebo::physics::DARTJoint::SetForce ( int *_index*, double *_effort* ) [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)). Forces are truncated by effortLimit before applied.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_effort* | Force value. |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.41.3.24 virtual void gazebo::physics::DARTJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [pure virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. |

Implemented in **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::DARTScrewJoint** (p. 344), **gazebo::physics::DARTBallJoint** (p. 283), **gazebo::physics::DARTSliderJoint** (p. 349), and **gazebo::physics::DARTUniversalJoint** (p. 356).

#### 10.41.3.25 virtual void gazebo::physics::DARTJoint::SetHighStop ( int *_index*, const math::Angle & *_angle* ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.41.3.26 virtual void gazebo::physics::DARTJoint::SetLowStop ( int *_index*, const math::Angle & *_angle* ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

### 10.41.4 Member Data Documentation

#### 10.41.4.1 DARTPhysicsPtr gazebo::physics::DARTJoint::dartPhysicsEngine [protected]

**DARTPhysics** (p. 330) engine pointer.

#### 10.41.4.2 dart::dynamics::BodyNode* gazebo::physics::DARTJoint::dtChildBodyNode [protected]

DART child body node pointer.

#### 10.41.4.3 dart::dynamics::Joint* gazebo::physics::DARTJoint::dtJoint [protected]

DART joint pointer.

The documentation for this class was generated from the following file:

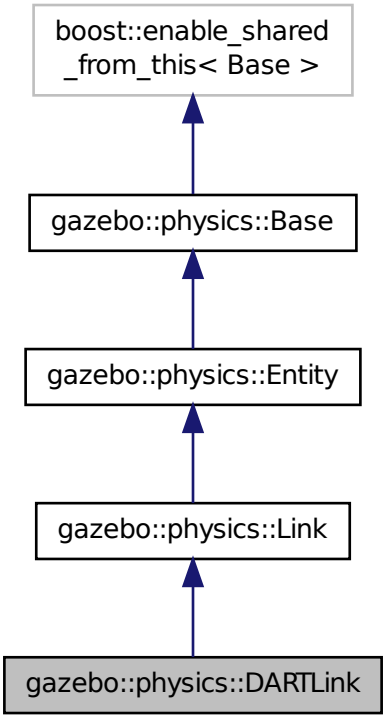
- **DARTJoint.hh**

## 10.42 gazebo::physics::DARTLink Class Reference

DART **Link** (p. 542) class.

`#include <DARTLink.hh>`

Inheritance diagram for gazebo::physics::DARTLink:




## Public Member Functions

- **DARTLink** (**EntityPtr** _parent)

  *Constructor.*
- virtual ~**DARTLink** ()

  *Destructor.*
- void **AddDARTChildJoint** (**DARTJointPtr** _dartChildJoint)

  *Set child joint of this link.*
- virtual void **AddForce** (const **math::Vector3** &_force)

  *Add a force to the body.*
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

  *Add a force to the body at position expressed to the body's own frame of reference.*
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

  *Add a force to the body using a global position.*
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

  *Add a force to the body, components are relative to the body's own frame of reference.*
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

  *Add a torque to the body, components are relative to the body's own frame of reference.*

- virtual void **AddTorque** (const **math::Vector3** &_torque)

  *Add a torque to the body.*
- virtual void **Fini** ()

  *Finalize the entity.*
- dart::dynamics::BodyNode * **GetDARTBodyNode** () const

  *Get pointer to DART BodyNode associated with this link.*
- **DARTModelPtr GetDARTModel** () const

  *Get pointer to DART **Model** (p. 624) associated with this link.*
- **DARTPhysicsPtr GetDARTPhysics** (void) const

  *Get pointer to DART Physics engine associated with this link.*
- dart::simulation::World * **GetDARTWorld** (void) const

  *Get pointer to DART **World** (p. 1157) associated with this link.*
- virtual bool **GetEnabled** () const

  *Get whether this body is enabled in the physics engine.*
- virtual bool **GetGravityMode** () const

  *Get the gravity mode.*
- virtual bool **GetKinematic** () const

  *Implement this function.*
- virtual **math::Vector3 GetWorldAngularVel** () const

  *Get the angular velocity of the entity in the world frame.*
- virtual **math::Vector3 GetWorldCoGLinearVel** () const

  *Get the linear velocity at the body's center of gravity in the world frame.*
- virtual **math::Vector3 GetWorldForce** () const

  *Get the force applied to the body in the world frame.*
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset=**math::Vector3**(0, 0, 0)) const

  *Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.*
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset, const **math::Quaternion** &_q) const

  *Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.*
- virtual **math::Vector3 GetWorldTorque** () const

  *Get the torque applied to the body in the world frame.*
- virtual void **Init** ()

  *Initialize the body.*
- virtual void **Load** (sdf::ElementPtr _ptr)

  *Load the body based on an SDF element.*
- virtual void **OnPoseChange** ()

  *This function is called when the entity's (or one of its parents) pose of the parent has changed.*
- virtual void **SetAngularDamping** (double _damping)

  *Set the angular damping factor.*
- virtual void **SetAngularVel** (const **math::Vector3** &_vel)

  *Set the angular velocity of the body.*
- virtual void **SetAutoDisable** (bool _disable)

  *Allow the link to auto disable.*
- void **SetDARTParentJoint** (**DARTJointPtr** _dartParentJoint)

  *Set parent joint of this link.*
- virtual void **SetEnabled** (bool _enable) const

  *Set whether this body is enabled.*
- virtual void **SetForce** (const **math::Vector3** &_force)

*Set the force applied to the body.*

- virtual void **SetGravityMode** (bool _mode)

  *Set whether gravity affects this body.*
- virtual void **SetKinematic** (const bool &_state)

  *Implement this function.*
- virtual void **SetLinearDamping** (double _damping)

  *Set the linear damping factor.*
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)

  *Set the linear velocity of the body.*
- virtual void **SetLinkStatic** (bool _static)

  *Freeze link to ground (inertial frame).*
- virtual void **SetSelfCollide** (bool _collide)

  *Set whether this body will collide with others in the model.*
- virtual void **SetTorque** (const **math::Vector3** &_torque)

  *Set the torque applied to the body.*
- void **updateDirtyPoseFromDARTTransformation** ()

  *Store DART Transformation to **Entity::dirtyPose*** (p. *389) and add this link to **World::dirtyPoses*** (p. *1169) so that World::Update() trigger **Entity::SetWorldPose()*** (p. *388) for this link.*

## Additional Inherited Members

### 10.42.1 Detailed Description

DART **Link** (p. 542) class.

### 10.42.2 Constructor & Destructor Documentation

#### 10.42.2.1 gazebo::physics::DARTLink::DARTLink ( EntityPtr *_parent* ) [explicit]

Constructor.

#### 10.42.2.2 virtual gazebo::physics::DARTLink::~DARTLink ( ) [virtual]

Destructor.

### 10.42.3 Member Function Documentation

#### 10.42.3.1 void gazebo::physics::DARTLink::AddDARTChildJoint ( DARTJointPtr *_dartChildJoint* )

Set child joint of this link.

**Parameters**

| | | |
|---|---|---|
| in | *_dartChildJoint* | Pointer to the child joint. |

#### 10.42.3.2 virtual void gazebo::physics::DARTLink::AddForce ( const math::Vector3 & _force ) [virtual]

Add a force to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |

Implements **gazebo::physics::Link** (p. 547).

#### 10.42.3.3 virtual void gazebo::physics::DARTLink::AddForceAtRelativePosition ( const math::Vector3 & _force, const math::Vector3 & _relPos ) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |
| in | *_relPos* | Position on the link to add the force. |

Implements **gazebo::physics::Link** (p. 547).

#### 10.42.3.4 virtual void gazebo::physics::DARTLink::AddForceAtWorldPosition ( const math::Vector3 & _force, const math::Vector3 & _pos ) [virtual]

Add a force to the body using a global position.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |
| in | *_pos* | Position in global coord frame to add the force. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.42.3.5 virtual void gazebo::physics::DARTLink::AddRelativeForce ( const math::Vector3 & _force ) [virtual]

Add a force to the body, components are relative to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.42.3.6 virtual void gazebo::physics::DARTLink::AddRelativeTorque ( const math::Vector3 & _torque ) [virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value to add. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.42.3.7 virtual void gazebo::physics::DARTLink::AddTorque ( const math::Vector3 & *_torque* ) [virtual]

Add a torque to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value to add to the link. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.42.3.8 virtual void gazebo::physics::DARTLink::Fini ( ) [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Entity** (p. 382).

#### 10.42.3.9 dart::dynamics::BodyNode* gazebo::physics::DARTLink::GetDARTBodyNode ( ) const

Get pointer to DART BodyNode associated with this link.

**Returns**

Pointer to DART BodyNode.

#### 10.42.3.10 DARTModelPtr gazebo::physics::DARTLink::GetDARTModel ( ) const

Get pointer to DART **Model** (p. 624) associated with this link.

**Returns**

Pointer to the DART **Model** (p. 624).

#### 10.42.3.11 DARTPhysicsPtr gazebo::physics::DARTLink::GetDARTPhysics ( void ) const

Get pointer to DART Physics engine associated with this link.

**Returns**

Pointer to the DART Physics engine.

#### 10.42.3.12 dart::simulation::World* gazebo::physics::DARTLink::GetDARTWorld ( void ) const

Get pointer to DART **World** (p. 1157) associated with this link.

**Returns**

Pointer to the DART **World** (p. 1157).

#### 10.42.3.13 virtual bool gazebo::physics::DARTLink::GetEnabled ( ) const [virtual]

Get whether this body is enabled in the physics engine.

**Returns**

True if the link is enabled.

Implements **gazebo::physics::Link** (p. 551).

#### 10.42.3.14 virtual bool gazebo::physics::DARTLink::GetGravityMode ( ) const [virtual]

Get the gravity mode.

**Returns**

True if gravity is enabled.

Implements **gazebo::physics::Link** (p. 551).

#### 10.42.3.15 virtual bool gazebo::physics::DARTLink::GetKinematic ( ) const [virtual]

Implement this function.

Get whether this body is in the kinematic state.

**Returns**

True if the link is kinematic only.

Reimplemented from **gazebo::physics::Link** (p. 552).

#### 10.42.3.16 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

**Returns**

A **math::Vector3** (p. 1091) for the velocity.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.42.3.17 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldCoGLinearVel ( ) const [virtual]

Get the linear velocity at the body's center of gravity in the world frame.

**Returns**

Linear velocity at the body's center of gravity in the world frame.

Implements **gazebo::physics::Link** (p. 555).

#### 10.42.3.18 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldForce ( ) const [virtual]

Get the force applied to the body in the world frame.

**Returns**

Force applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 555).

#### 10.42.3.19 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldLinearVel ( const math::Vector3 & *_offset* = math::Vector3(0, 0, 0) ) const [virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.

If no offset is given, the velocity at the origin of the **Link** (p. 542) frame will be returned.

**Parameters**

| | | |
|---|---|---|
| in | *_offset* | Offset of the point from the origin of the **Link** (p. 542) frame, expressed in the body-fixed frame. |

**Returns**

Linear velocity of the point on the body

Implements **gazebo::physics::Link** (p. 555).

#### 10.42.3.20 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldLinearVel ( const math::Vector3 & *_offset,* const math::Quaternion & *_q* ) const [virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

**Parameters**

| | | |
|---|---|---|
| in | *_offset* | Offset from the origin of the link frame expressed in a frame defined by _q. |
| in | *_q* | Describes the rotation of a reference frame relative to the world reference frame. |

**Returns**

Linear velocity of the point on the body in the world frame.

Implements **gazebo::physics::Link** (p. 556).

#### 10.42.3.21 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldTorque ( ) const [virtual]

Get the torque applied to the body in the world frame.

**Returns**

Torque applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 556).

#### 10.42.3.22 virtual void gazebo::physics::DARTLink::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Link** (p. 556).

#### 10.42.3.23 virtual void gazebo::physics::DARTLink::Load ( sdf::ElementPtr _*sdf* ) [virtual]

Load the body based on an SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters. |

Reimplemented from **gazebo::physics::Link** (p. 556).

#### 10.42.3.24 virtual void gazebo::physics::DARTLink::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Reimplemented from **gazebo::physics::Link** (p. 557).

#### 10.42.3.25 virtual void gazebo::physics::DARTLink::SetAngularDamping ( double _*damping* ) [virtual]

Set the angular damping factor.

**Parameters**

| | | |
|---|---|---|
| in | *_damping* | Angular damping factor. |

Implements **gazebo::physics::Link** (p. 558).

#### 10.42.3.26 virtual void gazebo::physics::DARTLink::SetAngularVel ( const math::Vector3 & _*vel* ) [virtual]

Set the angular velocity of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Angular velocity. |

Implements **gazebo::physics::Link** (p. 558).

#### 10.42.3.27 virtual void gazebo::physics::DARTLink::SetAutoDisable ( bool _*disable* ) [virtual]

Allow the link to auto disable.

**Parameters**

| | | |
|---|---|---|
| in | *_disable* | If true, the link is allowed to auto disable. |

Implements **gazebo::physics::Link** (p. 558).

#### 10.42.3.28 void gazebo::physics::DARTLink::SetDARTParentJoint ( DARTJointPtr _dartParentJoint )

Set parent joint of this link.

**Parameters**

| | | |
|---|---|---|
| in | *_dartParentJoint* | Pointer to the parent joint. |

#### 10.42.3.29 virtual void gazebo::physics::DARTLink::SetEnabled ( bool _enable ) const [virtual]

Set whether this body is enabled.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | True to enable the link in the physics engine. |

Implements **gazebo::physics::Link** (p. 559).

#### 10.42.3.30 virtual void gazebo::physics::DARTLink::SetForce ( const math::Vector3 & _force ) [virtual]

Set the force applied to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force value. |

Implements **gazebo::physics::Link** (p. 559).

#### 10.42.3.31 virtual void gazebo::physics::DARTLink::SetGravityMode ( bool _mode ) [virtual]

Set whether gravity affects this body.

**Parameters**

| | | |
|---|---|---|
| in | *_mode* | True to enable gravity. |

Implements **gazebo::physics::Link** (p. 559).

#### 10.42.3.32 virtual void gazebo::physics::DARTLink::SetKinematic ( const bool & _kinematic ) [virtual]

Implement this function.

Set whether this body is in the kinematic state.

**Parameters**

| | | |
|---|---|---|
| in | *_kinematic* | True to make the link kinematic only. |

Reimplemented from **gazebo::physics::Link** (p. 559).

#### 10.42.3.33 virtual void gazebo::physics::DARTLink::SetLinearDamping ( double _damping ) [virtual]

Set the linear damping factor.

**Parameters**

| | | |
|---|---|---|
| in | *_damping* | Linear damping factor. |

Implements **gazebo::physics::Link** (p. 560).

#### 10.42.3.34 virtual void gazebo::physics::DARTLink::SetLinearVel ( const math::Vector3 & _vel ) [virtual]

Set the linear velocity of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Linear velocity. |

Implements **gazebo::physics::Link** (p. 560).

#### 10.42.3.35 virtual void gazebo::physics::DARTLink::SetLinkStatic ( bool _static ) [virtual]

Freeze link to ground (inertial frame).

**Parameters**

| | | |
|---|---|---|
| in | *_static* | if true, freeze link to ground. Otherwise unfreeze link. |

Implements **gazebo::physics::Link** (p. 560).

#### 10.42.3.36 virtual void gazebo::physics::DARTLink::SetSelfCollide ( bool _collide ) [virtual]

Set whether this body will collide with others in the model.

**Parameters**

| | | |
|---|---|---|
| in | *_collid* | True to enable collisions. |

Implements **gazebo::physics::Link** (p. 561).

#### 10.42.3.37 virtual void gazebo::physics::DARTLink::SetTorque ( const math::Vector3 & _torque ) [virtual]

Set the torque applied to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value. |

Implements **gazebo::physics::Link** (p. 561).

#### 10.42.3.38 void gazebo::physics::DARTLink::updateDirtyPoseFromDARTTransformation ( )

Store DART Transformation to **Entity::dirtyPose** (p. 389) and add this link to **World::dirtyPoses** (p. 1169) so that World-::Update() trigger **Entity::SetWorldPose()** (p. 388) for this link.

The documentation for this class was generated from the following file:

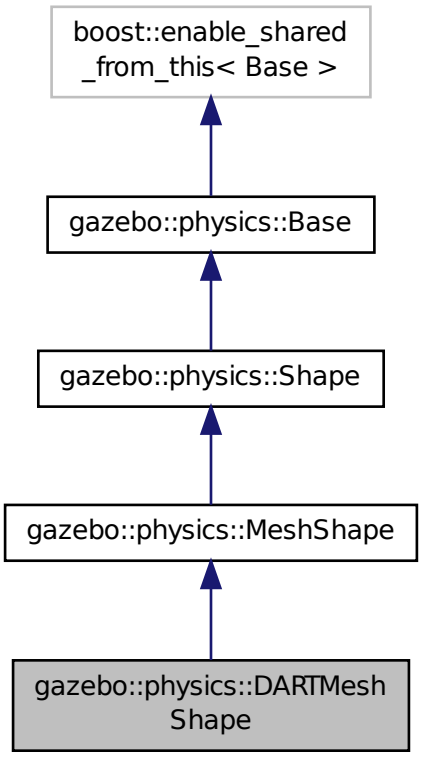
- **DARTLink.hh**

## 10.43 gazebo::physics::DARTMeshShape Class Reference

Triangle mesh collision.

`#include <DARTMeshShape.hh>`

Inheritance diagram for gazebo::physics::DARTMeshShape:




### Public Member Functions

- **DARTMeshShape** (**CollisionPtr** _parent)

  *Constructor.*

- virtual ~**DARTMeshShape** ()

  *Destructor.*
- virtual void **Init** ()

  *Initialize the shape.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load.*
- virtual void **Update** ()

  *Update the tri mesh.*

## Additional Inherited Members

### 10.43.1 Detailed Description

Triangle mesh collision.

### 10.43.2 Constructor & Destructor Documentation

#### 10.43.2.1 gazebo::physics::DARTMeshShape::DARTMeshShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent collision object. |

#### 10.43.2.2 virtual gazebo::physics::DARTMeshShape::~DARTMeshShape ( ) [virtual]

Destructor.

### 10.43.3 Member Function Documentation

#### 10.43.3.1 virtual void gazebo::physics::DARTMeshShape::Init ( ) [virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::MeshShape** (p. 623).

#### 10.43.3.2 virtual void gazebo::physics::DARTMeshShape::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load.

**Parameters**

| | | |
|---|---|---|
| in | *node* | Pointer to an SDF parameters |

Reimplemented from **gazebo::physics::Base** (p. 168).

#### 10.43.3.3 virtual void gazebo::physics::DARTMeshShape::Update ( ) [virtual]

Update the tri mesh.

Reimplemented from **gazebo::physics::MeshShape** (p. 624).

The documentation for this class was generated from the following file:

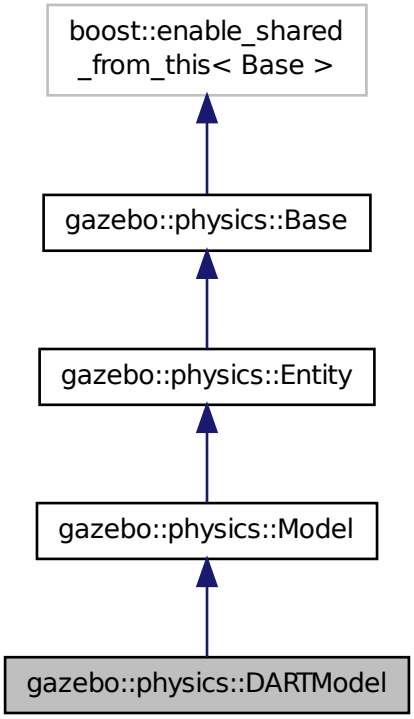
- **DARTMeshShape.hh**

## 10.44 gazebo::physics::DARTModel Class Reference

DART model class.

`#include <DARTModel.hh>`

Inheritance diagram for gazebo::physics::DARTModel:




### Public Member Functions

- **DARTModel** (**BasePtr** _parent)

  *Constructor.*

- virtual ~**DARTModel** ()

  *Destructor.*
- void **BackupState** ()
- virtual void **Fini** ()

  *Finalize the model.*
- **DARTPhysicsPtr GetDARTPhysics** (void) const
- dart::dynamics::Skeleton * **GetDARTSkeleton** ()
- dart::simulation::World * **GetDARTWorld** (void) const
- virtual void **Init** ()

  *Initialize the model.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the entity.*
- void **RestoreState** ()
- virtual void **Update** ()

  *Update the object.*

### Protected Attributes

- Eigen::VectorXd **dtConfig**
- dart::dynamics::Skeleton * **dtSkeleton**
- Eigen::VectorXd **dtVelocity**

### Additional Inherited Members

## 10.44.1 Detailed Description

DART model class.

## 10.44.2 Constructor & Destructor Documentation

#### 10.44.2.1 gazebo::physics::DARTModel::DARTModel ( **BasePtr** *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent object. |

#### 10.44.2.2 virtual gazebo::physics::DARTModel::~DARTModel ( ) [virtual]

Destructor.

## 10.44.3 Member Function Documentation

#### 10.44.3.1 void gazebo::physics::DARTModel::BackupState ( )

#### 10.44.3.2 virtual void gazebo::physics::DARTModel::Fini ( ) [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Model** (p. 629).

#### 10.44.3.3 DARTPhysicsPtr gazebo::physics::DARTModel::GetDARTPhysics ( void ) const

#### 10.44.3.4 dart::dynamics::Skeleton* gazebo::physics::DARTModel::GetDARTSkeleton ( )

#### 10.44.3.5 dart::simulation::World* gazebo::physics::DARTModel::GetDARTWorld ( void ) const

#### 10.44.3.6 virtual void gazebo::physics::DARTModel::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Model** (p. 633).

#### 10.44.3.7 virtual void gazebo::physics::DARTModel::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the entity.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to an SDF element. |

Reimplemented from **gazebo::physics::Entity** (p. 386).

#### 10.44.3.8 void gazebo::physics::DARTModel::RestoreState ( )

#### 10.44.3.9 virtual void gazebo::physics::DARTModel::Update ( ) [virtual]

Update the object.

Reimplemented from **gazebo::physics::Base** (p. 171).

### 10.44.4 Member Data Documentation

#### 10.44.4.1 Eigen::VectorXd gazebo::physics::DARTModel::dtConfig [protected]

#### 10.44.4.2 dart::dynamics::Skeleton* gazebo::physics::DARTModel::dtSkeleton [protected]

#### 10.44.4.3 Eigen::VectorXd gazebo::physics::DARTModel::dtVelocity [protected]

The documentation for this class was generated from the following file:

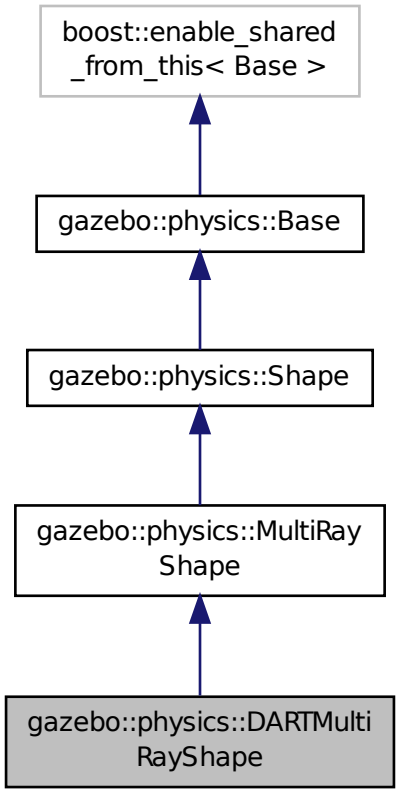
- **DARTModel.hh**

## 10.45 gazebo::physics::DARTMultiRayShape Class Reference

DART specific version of **MultiRayShape** (p. 664).

`#include <DARTMultiRayShape.hh>`

Inheritance diagram for gazebo::physics::DARTMultiRayShape:




## Public Member Functions

- **DARTMultiRayShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**DARTMultiRayShape** ()

  *Destructor.*
- virtual void **UpdateRays** ()

  *Physics engine specific method for updating the rays.*

## Protected Member Functions

- void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)

  *Add a ray to the collision.*

### Additional Inherited Members

#### 10.45.1 Detailed Description

DART specific version of **MultiRayShape** (p. 664).

The documentation for this class was generated from the following file:

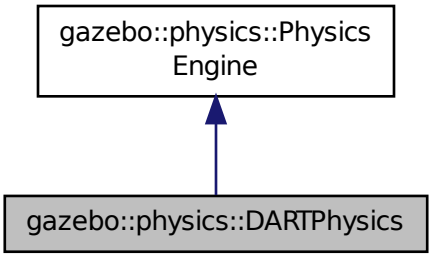
- **DARTMultiRayShape.hh**

## 10.46 gazebo::physics::DARTPhysics Class Reference

DART physics engine.

`#include <DARTPhysics.hh>`

Inheritance diagram for gazebo::physics::DARTPhysics:




### Public Types

- enum **DARTParam** { **MAX_CONTACTS**, **MIN_STEP_SIZE** }

  *DART physics parameter types.*

### Public Member Functions

- **DARTPhysics** (**WorldPtr** _world)

  *Constructor.*
- virtual ~**DARTPhysics** ()

  *Destructor.*
- virtual **CollisionPtr CreateCollision** (const std::string &_type, **LinkPtr** _body)

  *Create a collision.*
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)

  *Create a new joint.*
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)

*Create a new body.*

- virtual **ModelPtr CreateModel** (**BasePtr** _parent)

  *Create a new model.*

- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)

  *Create a **physics::Shape*** (p. *861) object.*

- virtual void **DebugPrint** () const

  *Debug print out of the physic engine state.*

- virtual void **Fini** ()

  *Finilize the physics engine.*

- dart::simulation::World * **GetDARTWorld** ()

  *Get pointer to DART **World*** (p. *1157) associated with this DART Physics.*

- virtual boost::any **GetParam** (const std::string &_key) const
- virtual boost::any **GetParam** (**DARTParam** _param) const

  *Get an parameter of the physics engine.*

- virtual std::string **GetType** () const

  *Return the physics engine type (ode|bullet|dart|simbody).*

- virtual void **Init** ()

  *Initialize the physics engine.*

- virtual void **InitForThread** ()

  *Init the engine for threads.*

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the physics engine.*

- virtual void **Reset** ()

  *Rest the physics engine.*

- virtual void **SetGravity** (const **gazebo::math::Vector3** &_gravity)

  *Set the gavity vector.*

- virtual void **SetSeed** (uint32_t _seed)

  *Set the random number seed for the physics engine.*

- virtual void **UpdateCollision** ()

  *Update the physics engine collision.*

- virtual void **UpdatePhysics** ()

  *Update the physics engine.*

## Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)

  *virtual callback for gztopic "~/physics".*

- virtual void **OnRequest** (ConstRequestPtr &_msg)

  *virtual callback for gztopic "~/request".*

## Additional Inherited Members

### 10.46.1 Detailed Description

DART physics engine.

### 10.46.2 Member Enumeration Documentation

#### 10.46.2.1 enum gazebo::physics::DARTPhysics::DARTParam

DART physics parameter types.

**Enumerator**

***MAX_CONTACTS*** Maximum number of contacts.

***MIN_STEP_SIZE*** Minimum step size.

### 10.46.3 Constructor & Destructor Documentation

#### 10.46.3.1 gazebo::physics::DARTPhysics::DARTPhysics ( WorldPtr *_world* )

Constructor.

#### 10.46.3.2 virtual gazebo::physics::DARTPhysics::~DARTPhysics ( ) [virtual]

Destructor.

### 10.46.4 Member Function Documentation

#### 10.46.4.1 virtual CollisionPtr gazebo::physics::DARTPhysics::CreateCollision ( const std::string & *_shapeType,* LinkPtr *_link* ) [virtual]

Create a collision.

**Parameters**

| | | |
|---|---|---|
| in | *_shapeType* | Type of collision to create. |
| in | *_link* | Parent link. |

Implements **gazebo::physics::PhysicsEngine** (p. 710).

#### 10.46.4.2 virtual JointPtr gazebo::physics::DARTPhysics::CreateJoint ( const std::string & *_type,* ModelPtr *_parent* ) [virtual]

Create a new joint.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | Type of joint to create. |
| in | *_parent* | **Model** (p. 624) parent. |

Implements **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.46.4.3 virtual LinkPtr gazebo::physics::DARTPhysics::CreateLink ( ModelPtr *_parent* ) [virtual]

Create a new body.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent model for the link. |

Implements **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.46.4.4 virtual ModelPtr gazebo::physics::DARTPhysics::CreateModel ( BasePtr *_base* ) [virtual]

Create a new model.

**Parameters**

| | | |
|---|---|---|
| in | *_base* | Boost shared pointer to a new model. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.46.4.5 virtual ShapePtr gazebo::physics::DARTPhysics::CreateShape ( const std::string & *_shapeType*, CollisionPtr *_collision* ) [virtual]

Create a **physics::Shape** (p. 861) object.

**Parameters**

| | | |
|---|---|---|
| in | *_shapeType* | Type of shape to create. |
| in | *_collision* | **Collision** (p. 220) parent. |

Implements **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.46.4.6 virtual void gazebo::physics::DARTPhysics::DebugPrint ( ) const [virtual]

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 712).

#### 10.46.4.7 virtual void gazebo::physics::DARTPhysics::Fini ( ) [virtual]

Finilize the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 712).

#### 10.46.4.8 dart::simulation::World* gazebo::physics::DARTPhysics::GetDARTWorld ( )

Get pointer to DART **World** (p. 1157) associated with this DART Physics.

**Returns**

The pointer to DART **World** (p. 1157).

#### 10.46.4.9 virtual boost::any gazebo::physics::DARTPhysics::GetParam ( const std::string & *_key* ) const [virtual]

#### 10.46.4.10 virtual boost::any gazebo::physics::DARTPhysics::GetParam ( DARTParam _param ) const [virtual]

Get an parameter of the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_param* | A parameter listed in the ODEParam enum |

**Returns**

The value of the parameter

#### 10.46.4.11 virtual std::string gazebo::physics::DARTPhysics::GetType ( ) const [virtual]

Return the physics engine type (ode|bullet|dart|simbody).

**Returns**

Type of the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 714).

#### 10.46.4.12 virtual void gazebo::physics::DARTPhysics::Init ( ) [virtual]

Initialize the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 715).

#### 10.46.4.13 virtual void gazebo::physics::DARTPhysics::InitForThread ( ) [virtual]

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 715).

#### 10.46.4.14 virtual void gazebo::physics::DARTPhysics::Load ( sdf::ElementPtr _sdf ) [virtual]

Load the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to the SDF parameters. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 715).

#### 10.46.4.15 virtual void gazebo::physics::DARTPhysics::OnPhysicsMsg ( ConstPhysicsPtr & _msg ) [protected], [virtual]

virtual callback for gztopic "~/physics".

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Physics message. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 716).

#### 10.46.4.16 virtual void gazebo::physics::DARTPhysics::OnRequest ( ConstRequestPtr & *_msg* ) [protected], [virtual]

virtual callback for gztopic "~/request".

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Request message. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 716).

#### 10.46.4.17 virtual void gazebo::physics::DARTPhysics::Reset ( ) [virtual]

Rest the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 716).

#### 10.46.4.18 virtual void gazebo::physics::DARTPhysics::SetGravity ( const gazebo::math::Vector3 & *_gravity* ) [virtual]

Set the gavity vector.

**Parameters**

| | | |
|---|---|---|
| in | *_gravity* | New gravity vector. |

Implements **gazebo::physics::PhysicsEngine** (p. 717).

#### 10.46.4.19 virtual void gazebo::physics::DARTPhysics::SetSeed ( uint32_t *_seed* ) [virtual]

Set the random number seed for the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_seed* | The random number seed. |

Implements **gazebo::physics::PhysicsEngine** (p. 718).

#### 10.46.4.20 virtual void gazebo::physics::DARTPhysics::UpdateCollision ( ) [virtual]

Update the physics engine collision.

Implements **gazebo::physics::PhysicsEngine** (p. 719).

#### 10.46.4.21 virtual void gazebo::physics::DARTPhysics::UpdatePhysics ( ) [virtual]

Update the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 719).

The documentation for this class was generated from the following file:

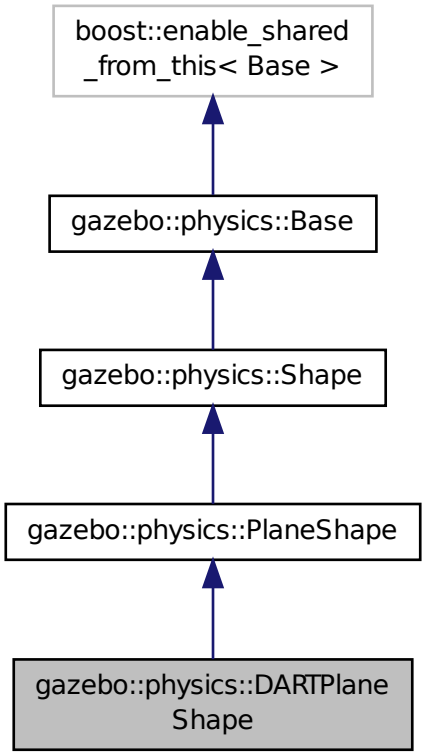
- **DARTPhysics.hh**

## 10.47 gazebo::physics::DARTPlaneShape Class Reference

An DART Plane shape.

`#include <DARTPlaneShape.hh>`

Inheritance diagram for gazebo::physics::DARTPlaneShape:




### Public Member Functions

- **DARTPlaneShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**DARTPlaneShape** ()

  *Destructor.*

- virtual void **CreatePlane** ()

  *Create the plane.*
- virtual void **SetAltitude** (const **math::Vector3** &_pos)

  *Set the altitude of the plane.*

### Additional Inherited Members

### 10.47.1 Detailed Description

An DART Plane shape.

### 10.47.2 Constructor & Destructor Documentation

#### 10.47.2.1 gazebo::physics::DARTPlaneShape::DARTPlaneShape ( CollisionPtr *_parent* ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent **Collision** (p. 220). |

#### 10.47.2.2 virtual gazebo::physics::DARTPlaneShape::~DARTPlaneShape ( ) [inline], [virtual]

Destructor.

### 10.47.3 Member Function Documentation

#### 10.47.3.1 virtual void gazebo::physics::DARTPlaneShape::CreatePlane ( ) [inline], [virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 730).

References gazebo::physics::Shape::collisionParent, gazebo::physics::PlaneShape::CreatePlane(), and gazebo::physics::DARTCollision::GetDARTBodyNode().

#### 10.47.3.2 virtual void gazebo::physics::DARTPlaneShape::SetAltitude ( const math::Vector3 & *_pos* ) [inline], [virtual]

Set the altitude of the plane.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | Position of the plane. |

Reimplemented from **gazebo::physics::PlaneShape** (p. 731).

References gazebo::physics::PlaneShape::SetAltitude().

The documentation for this class was generated from the following file:

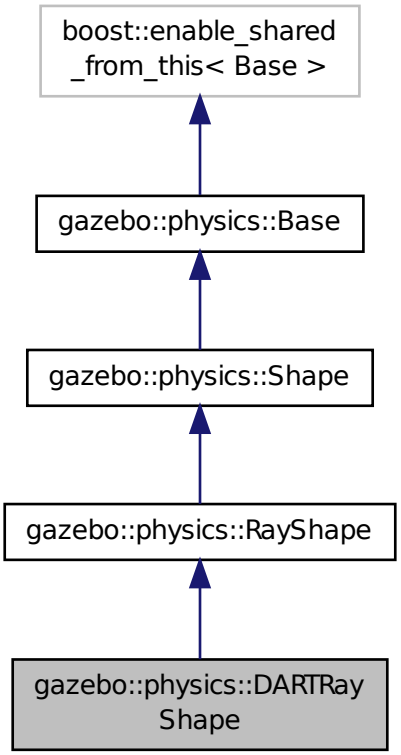
- **DARTPlaneShape.hh**

## 10.48 gazebo::physics::DARTRayShape Class Reference

Ray collision.

`#include <DARTRayShape.hh>`

Inheritance diagram for gazebo::physics::DARTRayShape:




### Public Member Functions

- **DARTRayShape** (**PhysicsEnginePtr** _physicsEngine)

  *Constructor for a global ray.*
- **DARTRayShape** (**CollisionPtr** _collision)

  *Constructor.*
- virtual ~**DARTRayShape** ()

  *Destructor.*
- virtual void **GetIntersection** (double &_dist, std::string &_entity)

*Get the nearest intersection.*

- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)

*Set the ray based on starting and ending points relative to the body.*

- virtual void **Update** ()

*Update the ray collision.*

### Additional Inherited Members

### 10.48.1 Detailed Description

Ray collision.

The documentation for this class was generated from the following file:

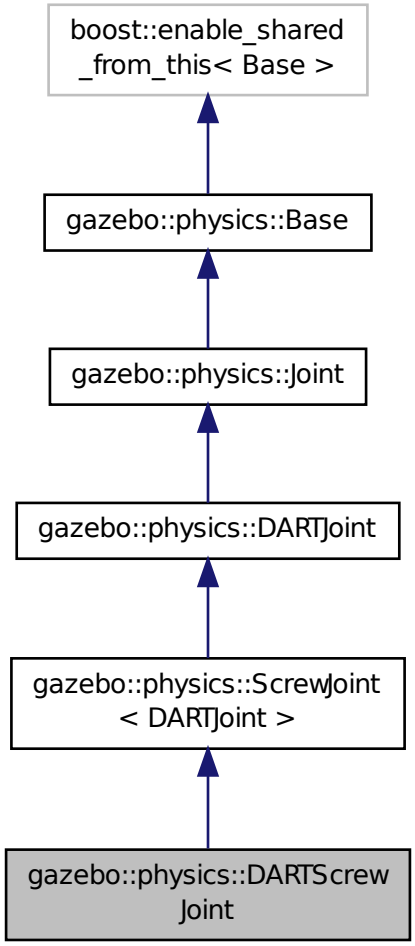
- **DARTRayShape.hh**

## 10.49 gazebo::physics::DARTScrewJoint Class Reference

A screw joint.

`#include <DARTScrewJoint.hh>`

Inheritance diagram for gazebo::physics::DARTScrewJoint:




## Public Member Functions

- **DARTScrewJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**DARTScrewJoint** ()

  *Destructor.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*

- virtual double **GetThreadPitch** (unsigned int _index)

  *Get screw joint thread pitch.*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **ScrewJoint** (p. 832).*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetMaxForce** (int _index, double _force)

  *Set the max allowed force of an axis(index).*
- virtual void **SetThreadPitch** (int _index, double _threadPitch)

  *Set screw joint thread pitch.*
- virtual void **SetVelocity** (int _index, double _vel)

  *Set the velocity of an axis(index).*

## Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 496).*

## Protected Attributes

- dart::dynamics::ScrewJoint * **dartScrewJoint**

  *Universal joint of DART.*

## Additional Inherited Members

### 10.49.1 Detailed Description

A screw joint.

### 10.49.2 Constructor & Destructor Documentation

#### 10.49.2.1 gazebo::physics::DARTScrewJoint::DARTScrewJoint ( BasePtr *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Pointer to the **Link** (p. 542) that is the joint' parent |

#### 10.49.2.2 virtual gazebo::physics::DARTScrewJoint::~DARTScrewJoint ( ) [virtual]

Destructor.

### 10.49.3 Member Function Documentation

#### 10.49.3.1 virtual math::Angle gazebo::physics::DARTScrewJoint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.49.3.2 virtual math::Vector3 gazebo::physics::DARTScrewJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.49.3.3 virtual double gazebo::physics::DARTScrewJoint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.49.3.4 virtual double gazebo::physics::DARTScrewJoint::GetThreadPitch ( unsigned int _index ) [virtual]

Get screw joint thread pitch.

This must be implemented in a child class

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

_threadPitch Thread pitch value.

Implements **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834).

#### 10.49.3.5 virtual double gazebo::physics::DARTScrewJoint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.49.3.6 virtual void gazebo::physics::DARTScrewJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 311).

#### 10.49.3.7 virtual void gazebo::physics::DARTScrewJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load a **ScrewJoint** (p. 832).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF value to load from |

Reimplemented from **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834).

#### 10.49.3.8 virtual void gazebo::physics::DARTScrewJoint::SetAxis ( int *_index*, const math::Vector3 & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.49.3.9 virtual void gazebo::physics::DARTScrewJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. |

Implements **gazebo::physics::DARTJoint** (p. 312).

#### 10.49.3.10 virtual void gazebo::physics::DARTScrewJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.49.3.11 virtual void gazebo::physics::DARTScrewJoint::SetThreadPitch ( int *_index*, double *_threadPitch* ) [virtual]

Set screw joint thread pitch.

This must be implemented in a child class

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_threadPitch* | Thread pitch value. |

Implements **gazebo::physics::ScrewJoint**< **DARTJoint** > (p. 835).

#### 10.49.3.12 virtual void gazebo::physics::DARTScrewJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

### 10.49.4 Member Data Documentation

#### 10.49.4.1 dart::dynamics::ScrewJoint* gazebo::physics::DARTScrewJoint::dartScrewJoint [protected]

Universal joint of DART.

The documentation for this class was generated from the following file:

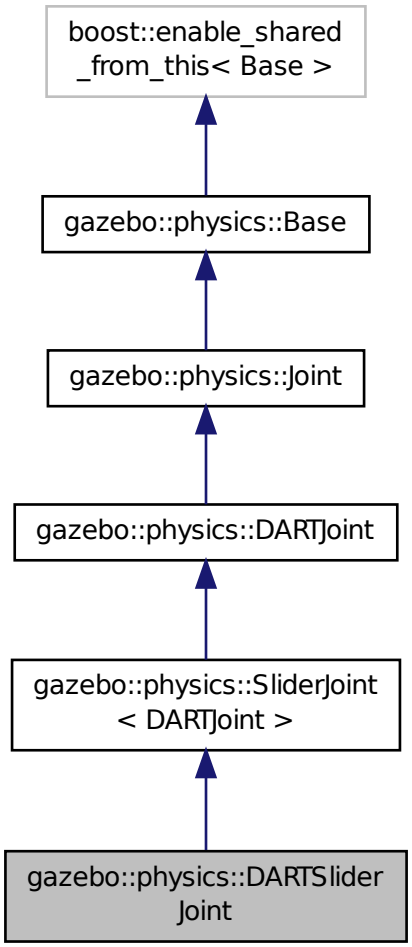
- **DARTScrewJoint.hh**

## 10.50 gazebo::physics::DARTSliderJoint Class Reference

A slider joint.

`#include <DARTSliderJoint.hh>`

Inheritance diagram for gazebo::physics::DARTSliderJoint:



### Public Member Functions

- **DARTSliderJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**DARTSliderJoint** ()

  *Destructor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **SliderJoint** (p. 973).*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetMaxForce** (int _index, double _force)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _vel)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 496).*

### Protected Attributes

- dart::dynamics::PrismaticJoint * **dtPrismaticJoint**

  *Prismatic joint of DART.*

### Additional Inherited Members

### 10.50.1 Detailed Description

A slider joint.

## 10.50.2 Constructor & Destructor Documentation

### 10.50.2.1 gazebo::physics::DARTSliderJoint::DARTSliderJoint ( BasePtr _parent )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Pointer to the **Link** (p. 542) that is the joint' parent |

### 10.50.2.2 virtual gazebo::physics::DARTSliderJoint::~DARTSliderJoint ( ) [virtual]

Destructor.

## 10.50.3 Member Function Documentation

### 10.50.3.1 virtual math::Vector3 gazebo::physics::DARTSliderJoint::GetAnchor ( int _index ) const [virtual]

Get the anchor.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. Not used. |

**Returns**

Anchor for the joint.

Reimplemented from **gazebo::physics::SliderJoint< DARTJoint >** (p. 974).

### 10.50.3.2 virtual math::Angle gazebo::physics::DARTSliderJoint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

### 10.50.3.3 virtual math::Vector3 gazebo::physics::DARTSliderJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.50.3.4 virtual double gazebo::physics::DARTSliderJoint::GetMaxForce ( int *_index* ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.50.3.5 virtual double gazebo::physics::DARTSliderJoint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.50.3.6 virtual void gazebo::physics::DARTSliderJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 311).

#### 10.50.3.7 virtual void gazebo::physics::DARTSliderJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load a **SliderJoint** (p. 973).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from |

Reimplemented from **gazebo::physics::SliderJoint**< **DARTJoint** > (p. 975).

#### 10.50.3.8 virtual void gazebo::physics::DARTSliderJoint::SetAxis ( int *_index*, const math::Vector3 & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.50.3.9 virtual void gazebo::physics::DARTSliderJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. |

Implements **gazebo::physics::DARTJoint** (p. 312).

#### 10.50.3.10 virtual void gazebo::physics::DARTSliderJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.50.3.11 virtual void gazebo::physics::DARTSliderJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

### 10.50.4 Member Data Documentation

#### 10.50.4.1 dart::dynamics::PrismaticJoint* gazebo::physics::DARTSliderJoint::dtPrismaticJoint [protected]

Prismatic joint of DART.

The documentation for this class was generated from the following file:

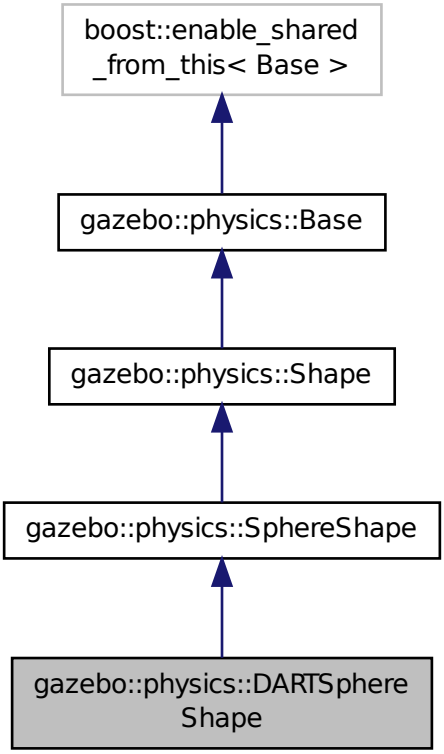
- **DARTSliderJoint.hh**

## 10.51 gazebo::physics::DARTSphereShape Class Reference

A DART sphere shape.

`#include <DARTSphereShape.hh>`

Inheritance diagram for gazebo::physics::DARTSphereShape:




### Public Member Functions

- **DARTSphereShape** (**DARTCollisionPtr** _parent)

  *Constructor.*
- virtual ~**DARTSphereShape** ()

*Destructor.*

- virtual void **SetRadius** (double _radius)

  *Set the size.*

### Additional Inherited Members

### 10.51.1 Detailed Description

A DART sphere shape.

### 10.51.2 Constructor & Destructor Documentation

#### 10.51.2.1 gazebo::physics::DARTSphereShape::DARTSphereShape ( DARTCollisionPtr *_parent* ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent **Collision** (p. 220). |

#### 10.51.2.2 virtual gazebo::physics::DARTSphereShape::~DARTSphereShape ( ) [inline], [virtual]

Destructor.

### 10.51.3 Member Function Documentation

#### 10.51.3.1 virtual void gazebo::physics::DARTSphereShape::SetRadius ( double *_radius* ) [inline], [virtual]

Set the size.

**Parameters**

| | | |
|---|---|---|
| in | *_radius* | Radius of the sphere. |

Reimplemented from **gazebo::physics::SphereShape** (p. 988).

References gazebo::physics::Shape::collisionParent, gazebo::math::equal(), gazebo::physics::DARTCollision::GetDARTBodyNode(), gzerr, gzwarn, NULL, and gazebo::physics::SphereShape::SetRadius().

The documentation for this class was generated from the following file:

- **DARTSphereShape.hh**

## 10.52 gazebo::physics::DARTTypes Class Reference

A set of functions for converting between the math types used by gazebo and dart.

`#include <DARTTypes.hh>`

### Static Public Member Functions

- static Eigen::Isometry3d **ConvPose** (const **math::Pose** &_pose)
- static **math::Pose ConvPose** (const Eigen::Isometry3d &_T)
- static Eigen::Quaterniond **ConvQuat** (const **math::Quaternion** &_quat)
- static **math::Quaternion ConvQuat** (const Eigen::Quaterniond &_quat)
- static Eigen::Vector3d **ConvVec3** (const **math::Vector3** &_vec3)
- static **math::Vector3 ConvVec3** (const Eigen::Vector3d &_vec3)

#### 10.52.1 Detailed Description

A set of functions for converting between the math types used by gazebo and dart.

The documentation for this class was generated from the following file:

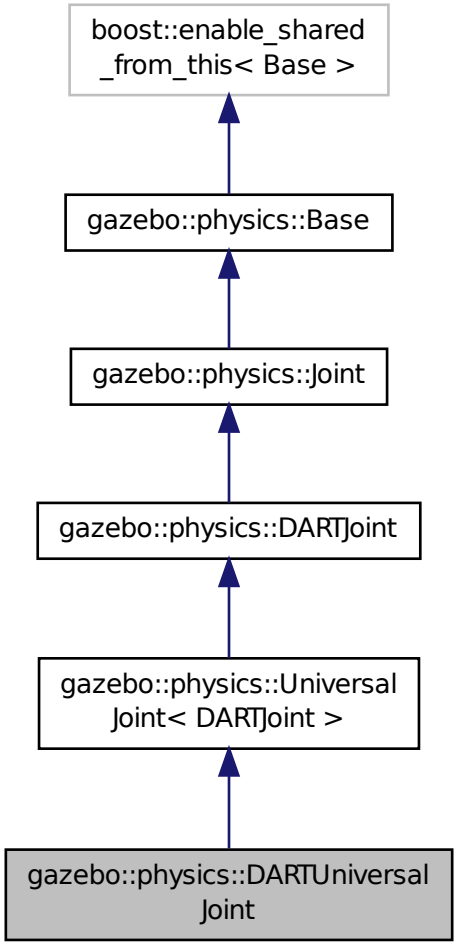
- **DARTTypes.hh**

## 10.53 gazebo::physics::DARTUniversalJoint Class Reference

A universal joint.

`#include <DARTUniversalJoint.hh>`

Inheritance diagram for gazebo::physics::DARTUniversalJoint:




## Public Member Functions

- **DARTUniversalJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**DARTUniversalJoint** ()

  *Destuctor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*

- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **UniversalJoint** (p. 1064).*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetMaxForce** (int _index, double _force)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _vel)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 496).*

### Protected Attributes

- dart::dynamics::UniversalJoint * **dtUniveralJoint**

  *Universal joint of DART.*

### Additional Inherited Members

## 10.53.1 Detailed Description

A universal joint.

## 10.53.2 Constructor & Destructor Documentation

#### 10.53.2.1 gazebo::physics::DARTUniversalJoint::DARTUniversalJoint ( **BasePtr** *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Pointer to the **Link** (p. 542) that is the joint' parent |

#### 10.53.2.2 virtual gazebo::physics::DARTUniversalJoint::~DARTUniversalJoint ( ) [virtual]

Destuctor.

### 10.53.3 Member Function Documentation

#### 10.53.3.1 virtual math::Vector3 gazebo::physics::DARTUniversalJoint::GetAnchor ( int _index ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 502).

#### 10.53.3.2 virtual math::Angle gazebo::physics::DARTUniversalJoint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.53.3.3 virtual math::Vector3 gazebo::physics::DARTUniversalJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.53.3.4 virtual double gazebo::physics::DARTUniversalJoint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.53.3.5 virtual double gazebo::physics::DARTUniversalJoint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.53.3.6 virtual void gazebo::physics::DARTUniversalJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 311).

#### 10.53.3.7 virtual void gazebo::physics::DARTUniversalJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load a **UniversalJoint** (p. 1064).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1065).

#### 10.53.3.8 virtual void gazebo::physics::DARTUniversalJoint::SetAxis ( int *_index*, const **math::Vector3** & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.53.3.9 virtual void gazebo::physics::DARTUniversalJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. |

Implements **gazebo::physics::DARTJoint** (p. 312).

#### 10.53.3.10 virtual void gazebo::physics::DARTUniversalJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.53.3.11 virtual void gazebo::physics::DARTUniversalJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

### 10.53.4 Member Data Documentation

#### 10.53.4.1 dart::dynamics::UniversalJoint* gazebo::physics::DARTUniversalJoint::dtUniveralJoint [protected]

Universal joint of DART.

The documentation for this class was generated from the following file:

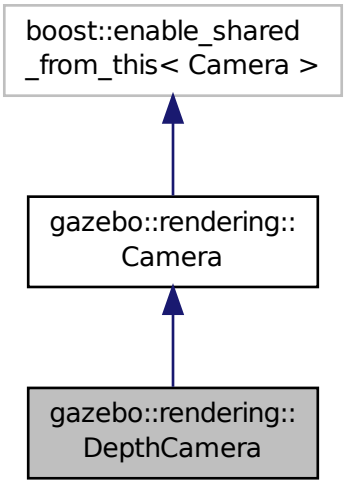
- **DARTUniversalJoint.hh**

## 10.54 gazebo::rendering::DepthCamera Class Reference

Depth camera used to render depth data into an image buffer.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DepthCamera:




## Public Member Functions

- **DepthCamera** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)

  *Constructor.*
- virtual ~**DepthCamera** ()

  *Destructor.*
- template<typename T >
  **event::ConnectionPtr ConnectNewDepthFrame** (T _subscriber)

  *Connect a to the new depth image signal.*
- template<typename T >
  **event::ConnectionPtr ConnectNewRGBPointCloud** (T _subscriber)

  *Connect a to the new rgb point cloud signal.*
- void **CreateDepthTexture** (const std::string &_textureName)

  *Create a texture which will hold the depth data.*
- void **DisconnectNewDepthFrame** (**event::ConnectionPtr** &_c)

  *Disconnect from an depth image singal.*
- void **DisconnectNewRGBPointCloud** (**event::ConnectionPtr** &c)

  *Disconnect from an rgb point cloud singal.*
- void **Fini** ()

  *Finalize the camera.*
- virtual const float * **GetDepthData** ()

  *All things needed to get back z buffer for depth data.*
- void **Init** ()

  *Initialize the camera.*
- void **Load** (sdf::ElementPtr &_sdf)

*Load the camera with a set of parmeters.*

- void **Load** ()

  *Load the camera with default parmeters.*

- virtual void **PostRender** ()

  *Render the camera.*

- virtual void **SetDepthTarget** (Ogre::RenderTarget ∗_target)

  *Set the render target, which renders the depth data.*

### Protected Attributes

- Ogre::RenderTarget ∗ **depthTarget**

  *Pointer to the depth target.*

- Ogre::Texture ∗ **depthTexture**

  *Pointer to the depth texture.*

- Ogre::Viewport ∗ **depthViewport**

  *Pointer to the depth viewport.*

### Additional Inherited Members

## 10.54.1 Detailed Description

Depth camera used to render depth data into an image buffer.

## 10.54.2 Constructor & Destructor Documentation

### 10.54.2.1 gazebo::rendering::DepthCamera::DepthCamera ( const std::string & *_namePrefix,* ScenePtr *_scene,* bool *_autoRender* = `true` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_namePrefix* | Unique prefix name for the camera. |
| in | *_scene* | **Scene** (p. 814) that will contain the camera |
| in | *_autoRender* | Almost everyone should leave this as true. |

### 10.54.2.2 virtual gazebo::rendering::DepthCamera::∼DepthCamera ( ) `[virtual]`

Destructor.

## 10.54.3 Member Function Documentation

### 10.54.3.1 template<typename T > event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewDepthFrame ( T *_subscriber* ) `[inline]`

Connect a to the new depth image signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Subscriber callback function |

**Returns**

Pointer to the new Connection. This must be kept in scope

References gazebo::event::EventT< T >::Connect().

#### 10.54.3.2 template<typename T > event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud ( T *_subscriber* ) [inline]

Connect a to the new rgb point cloud signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Subscriber callback function |

**Returns**

Pointer to the new Connection. This must be kept in scope

References gazebo::event::EventT< T >::Connect().

#### 10.54.3.3 void gazebo::rendering::DepthCamera::CreateDepthTexture ( const std::string & *_textureName* )

Create a texture which will hold the depth data.

**Parameters**

| | | |
|---|---|---|
| in | *_textureName* | Name of the texture to create |

#### 10.54.3.4 void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame ( event::ConnectionPtr & *_c* ) [inline]

Disconnect from an depth image singal.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The connection to disconnect |

References gazebo::event::EventT< T >::Disconnect().

#### 10.54.3.5 void gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud ( event::ConnectionPtr & *c* ) [inline]

Disconnect from an rgb point cloud singal.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The connection to disconnect |

References gazebo::event::EventT< T >::Disconnect().

#### 10.54.3.6 void gazebo::rendering::DepthCamera::Fini ( ) [virtual]

Finalize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 195).

#### 10.54.3.7 virtual const float* gazebo::rendering::DepthCamera::GetDepthData ( ) [virtual]

All things needed to get back z buffer for depth data.

**Returns**

The z-buffer as a float array

#### 10.54.3.8 void gazebo::rendering::DepthCamera::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 203).

#### 10.54.3.9 void gazebo::rendering::DepthCamera::Load ( sdf::ElementPtr & *_sdf* )

Load the camera with a set of parmeters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | The SDF camera info |

#### 10.54.3.10 void gazebo::rendering::DepthCamera::Load ( ) [virtual]

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.54.3.11 virtual void gazebo::rendering::DepthCamera::PostRender ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.54.3.12 virtual void gazebo::rendering::DepthCamera::SetDepthTarget ( Ogre::RenderTarget * *_target* ) [virtual]

Set the render target, which renders the depth data.

**Parameters**

| | | |
|---|---|---|
| in | *_target* | Pointer to the render target |

### 10.54.4 Member Data Documentation

#### 10.54.4.1 Ogre::RenderTarget* gazebo::rendering::DepthCamera::depthTarget [protected]

Pointer to the depth target.

#### 10.54.4.2 Ogre::Texture* gazebo::rendering::DepthCamera::depthTexture [protected]

Pointer to the depth texture.

#### 10.54.4.3 Ogre::Viewport* gazebo::rendering::DepthCamera::depthViewport [protected]

Pointer to the depth viewport.

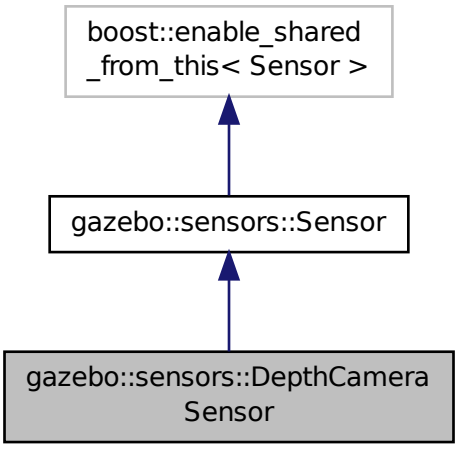
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

## 10.55 gazebo::sensors::DepthCameraSensor Class Reference

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::DepthCameraSensor:




### Public Member Functions

- **DepthCameraSensor** ()

  *Constructor.*

- virtual ~**DepthCameraSensor** ()
  *Destructor.*
- **rendering::DepthCameraPtr GetDepthCamera** () const
  *Returns a pointer to the **rendering::DepthCamera** (p. 357).*
- bool **SaveFrame** (const std::string &_filename)
  *Saves an image frame of depth camera sensor to file.*
- virtual void **SetActive** (bool _value)
  *Set whether the sensor is active or not.*

### Protected Member Functions

- virtual void **Fini** ()
  *Finalize the camera.*
- virtual void **Init** ()
  *Initialize the camera.*
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr &_sdf)
  *Load the sensor with SDF parameters.*
- virtual void **Load** (const std::string &_worldName)
  *Load the sensor with default parameters.*
- virtual void **UpdateImpl** (bool _force)
  *Update the sensor information.*

### Additional Inherited Members

## 10.55.1 Constructor & Destructor Documentation

#### 10.55.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ( )

Constructor.

#### 10.55.1.2 virtual gazebo::sensors::DepthCameraSensor::~DepthCameraSensor ( ) [virtual]

Destructor.

## 10.55.2 Member Function Documentation

#### 10.55.2.1 virtual void gazebo::sensors::DepthCameraSensor::Fini ( ) [protected], [virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.55.2.2 rendering::DepthCameraPtr gazebo::sensors::DepthCameraSensor::GetDepthCamera ( ) const [inline]

Returns a pointer to the **rendering::DepthCamera** (p. 357).

**Returns**

Depth Camera pointer

#### 10.55.2.3 virtual void gazebo::sensors::DepthCameraSensor::Init ( ) [protected], [virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.55.2.4 virtual void gazebo::sensors::DepthCameraSensor::Load ( const std::string & *_worldName,* sdf::ElementPtr & *_sdf* ) [protected], [virtual]

Load the sensor with SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF **Sensor** (p. 837) parameters |
| in | *_worldName* | Name of world to load from |

#### 10.55.2.5 virtual void gazebo::sensors::DepthCameraSensor::Load ( const std::string & *_worldName* ) [protected], [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.55.2.6 bool gazebo::sensors::DepthCameraSensor::SaveFrame ( const std::string & *_filename* )

Saves an image frame of depth camera sensor to file.

**Parameters**

| | | |
|---|---|---|
| in | *Name* | of file to save as |

**Returns**

True if saved, false if not

#### 10.55.2.7 virtual void gazebo::sensors::DepthCameraSensor::SetActive ( bool *_value* ) [virtual]

Set whether the sensor is active or not.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | True if active, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.55.2.8 virtual void gazebo::sensors::DepthCameraSensor::UpdateImpl ( bool *_force* ) [protected], [virtual]

Update the sensor information.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

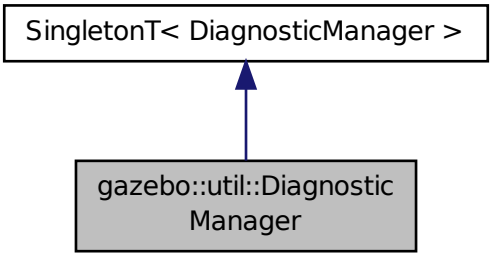
- **DepthCameraSensor.hh**

## 10.56 gazebo::util::DiagnosticManager Class Reference

A diagnostic manager class.

`#include <util/util.hh>`

Inheritance diagram for gazebo::util::DiagnosticManager:




### Public Member Functions

- std::string **GetLabel** (int _index) const

  *Get a label for a timer.*
- boost::filesystem::path **GetLogPath** () const

  *Get the path in which logs are stored.*
- **common::Time GetTime** (int _index) const

  *Get the time of a timer instance.*
- **common::Time GetTime** (const std::string &_label) const

  *Get a time based on a label.*
- int **GetTimerCount** () const

  *Get the number of timers.*
- void **Init** (const std::string &_worldName)

*Initialize to report diagnostics about a world.*

- void **Lap** (const std::string &_name, const std::string &_prefix)

  *Output the current elapsed time of an active timer with a prefix string.*
- void **StartTimer** (const std::string &_name)

  *Start a new timer instance.*
- void **StopTimer** (const std::string &_name)

  *Stop a currently running timer.*

## Additional Inherited Members

### 10.56.1 Detailed Description

A diagnostic manager class.

### 10.56.2 Member Function Documentation

#### 10.56.2.1 std::string gazebo::util::DiagnosticManager::GetLabel ( int *_index* ) const

Get a label for a timer.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of a timer instance |

**Returns**

Label of the specified timer

#### 10.56.2.2 boost::filesystem::path gazebo::util::DiagnosticManager::GetLogPath ( ) const

Get the path in which logs are stored.

**Returns**

The path in which logs are stored.

#### 10.56.2.3 common::Time gazebo::util::DiagnosticManager::GetTime ( int *_index* ) const

Get the time of a timer instance.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | The index of a timer instance |

**Returns**

Time of the specified timer

#### 10.56.2.4 common::Time gazebo::util::DiagnosticManager::GetTime ( const std::string & _label ) const

Get a time based on a label.

**Parameters**

| | | |
|---|---|---|
| in | *_label* | Name of the timer instance |

**Returns**

Time of the specified timer

#### 10.56.2.5 int gazebo::util::DiagnosticManager::GetTimerCount ( ) const

Get the number of timers.

**Returns**

The number of timers

#### 10.56.2.6 void gazebo::util::DiagnosticManager::Init ( const std::string & _worldName )

Initialize to report diagnostics about a world.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of the world. |

#### 10.56.2.7 void gazebo::util::DiagnosticManager::Lap ( const std::string & _name, const std::string & _prefix )

Output the current elapsed time of an active timer with a prefix string.

This also resets the timer and keeps it running.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the timer to access. |
| in | *_prefix* | Informational string that is output with the elapsed time. |

#### 10.56.2.8 void gazebo::util::DiagnosticManager::StartTimer ( const std::string & _name )

Start a new timer instance.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the timer. |

**Returns**

A pointer to the new diagnostic timer

#### 10.56.2.9 void gazebo::util::DiagnosticManager::StopTimer ( const std::string & _name )

Stop a currently running timer.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the timer to stop. |

The documentation for this class was generated from the following file:

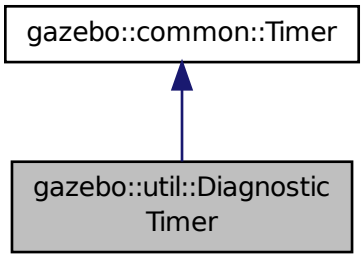
- **Diagnostics.hh**

## 10.57 gazebo::util::DiagnosticTimer Class Reference

A timer designed for diagnostics.

`#include <util/util.hh>`

Inheritance diagram for gazebo::util::DiagnosticTimer:




### Public Member Functions

- **DiagnosticTimer** (const std::string &_name)

  *Constructor.*
- virtual ~**DiagnosticTimer** ()

  *Destructor.*
- const std::string **GetName** () const

  *Get the name of the timer.*
- void **Lap** (const std::string &_prefix)

  *Output a lap time.*
- virtual void **Start** ()

  *Start the timer.*
- virtual void **Stop** ()

  *Stop the timer.*

### 10.57.1 Detailed Description

A timer designed for diagnostics.

### 10.57.2 Constructor & Destructor Documentation

#### 10.57.2.1 gazebo::util::DiagnosticTimer::DiagnosticTimer ( const std::string & *_name* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the timer |

#### 10.57.2.2 virtual gazebo::util::DiagnosticTimer::~DiagnosticTimer ( ) [virtual]

Destructor.

### 10.57.3 Member Function Documentation

#### 10.57.3.1 const std::string gazebo::util::DiagnosticTimer::GetName ( ) const [inline]

Get the name of the timer.

**Returns**

The name of timer

#### 10.57.3.2 void gazebo::util::DiagnosticTimer::Lap ( const std::string & *_prefix* )

Output a lap time.

**Parameters**

| | | |
|---|---|---|
| in | *_prefix* | Annotation to output with the elapsed time. |

#### 10.57.3.3 virtual void gazebo::util::DiagnosticTimer::Start ( ) [virtual]

Start the timer.

Reimplemented from **gazebo::common::Timer** (p. 1054).

#### 10.57.3.4 virtual void gazebo::util::DiagnosticTimer::Stop ( ) [virtual]

Stop the timer.

Reimplemented from **gazebo::common::Timer** (p. 1054).

The documentation for this class was generated from the following file:

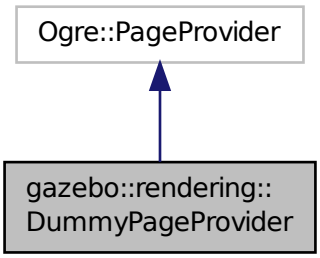
- **Diagnostics.hh**

## 10.58 gazebo::rendering::DummyPageProvider Class Reference

Pretends to provide procedural page content to avoid page loading.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DummyPageProvider:




### Public Member Functions

- bool **loadProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)

  *Give a provider the opportunity to load page content procedurally.*
- bool **prepareProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)

  *Give a provider the opportunity to prepare page content procedurally.*
- bool **unloadProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)

  *Give a provider the opportunity to unload page content procedurally.*
- bool **unprepareProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)

  *Give a provider the opportunity to unprepare page content procedurally.*

### 10.58.1 Detailed Description

Pretends to provide procedural page content to avoid page loading.

### 10.58.2 Member Function Documentation

#### 10.58.2.1 bool gazebo::rendering::DummyPageProvider::loadProceduralPage ( Ogre::Page *, Ogre::PagedWorldSection * ) [inline]

Give a provider the opportunity to load page content procedurally.

The parameters are not used.

#### 10.58.2.2 bool gazebo::rendering::DummyPageProvider::prepareProceduralPage ( Ogre::Page * , Ogre::PagedWorldSection * ) [inline]

Give a provider the opportunity to prepare page content procedurally.

The parameters are not used.

#### 10.58.2.3 bool gazebo::rendering::DummyPageProvider::unloadProceduralPage ( Ogre::Page * , Ogre::PagedWorldSection * ) [inline]

Give a provider the opportunity to unload page content procedurally.

The parameters are not used.

#### 10.58.2.4 bool gazebo::rendering::DummyPageProvider::unprepareProceduralPage ( Ogre::Page * , Ogre::PagedWorldSection * ) [inline]

Give a provider the opportunity to unprepare page content procedurally.

The parameters are not used.

The documentation for this class was generated from the following file:

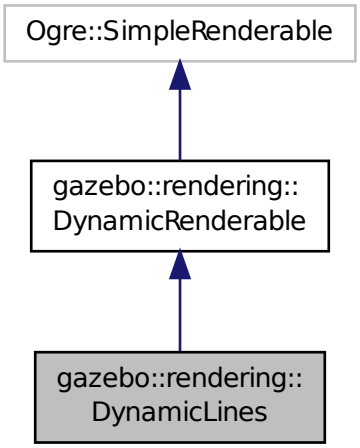
- **Heightmap.hh**

## 10.59 gazebo::rendering::DynamicLines Class Reference

Class for drawing lines that can change.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DynamicLines:



## Public Member Functions

- **DynamicLines** (**RenderOpType** _opType=**RENDERING_LINE_STRIP**)

  *Constructor.*
- virtual ~**DynamicLines** ()

  *Destructor.*
- void **AddPoint** (const **math::Vector3** &_pt, const **common::Color** &_color=**common::Color::White**)

  *Add a point to the point list.*
- void **AddPoint** (double _x, double _y, double _z, const **common::Color** &_color=**common::Color::White**)

  *Add a point to the point list.*
- void **Clear** ()

  *Remove all points from the point list.*
- virtual const Ogre::String & **getMovableType** () const

  *Overridden function from **Ogre** (p. 130)'s base class.*
- const **math::Vector3** & **GetPoint** (unsigned int _index) const

  *Return the location of an existing point in the point list.*
- unsigned int **GetPointCount** () const

  *Return the total number of points in the point list.*
- void **SetColor** (unsigned int _index, const **common::Color** &_color)

  *Change the color of an existing point in the point list.*
- void **SetPoint** (unsigned int _index, const **math::Vector3** &_value)

  *Change the location of an existing point in the point list.*
- void **Update** ()

  *Call this to update the hardware buffer after making changes.*

## Static Public Member Functions

- static std::string **GetMovableType** ()

  *Get type of movable.*

## Additional Inherited Members

### 10.59.1 Detailed Description

Class for drawing lines that can change.

### 10.59.2 Constructor & Destructor Documentation

#### 10.59.2.1 gazebo::rendering::DynamicLines::DynamicLines ( RenderOpType _opType = RENDERING_LINE_STRIP )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_opType* | The type of Line |

#### 10.59.2.2 virtual gazebo::rendering::DynamicLines::~DynamicLines ( ) [virtual]

Destructor.

### 10.59.3 Member Function Documentation

#### 10.59.3.1 void gazebo::rendering::DynamicLines::AddPoint ( const math::Vector3 & _pt, const common::Color & _color = common::Color::White )

Add a point to the point list.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | **math::Vector3** (p. 1091) point |
| in | *_color* | **common::Color** (p. 233) Point color |

#### 10.59.3.2 void gazebo::rendering::DynamicLines::AddPoint ( double _x, double _y, double _z, const common::Color & _color = common::Color::White )

Add a point to the point list.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | X position |
| in | *_y* | Y position |
| in | *_z* | Z position |
| in | *_color* | **common::Color** (p. 233) Point color |

#### 10.59.3.3 void gazebo::rendering::DynamicLines::Clear ( )

Remove all points from the point list.

#### 10.59.3.4 static std::string gazebo::rendering::DynamicLines::GetMovableType ( ) [static]

Get type of movable.

**Returns**

This returns "gazebo::dynamiclines"

#### 10.59.3.5 virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType ( ) const [virtual]

Overridden function from **Ogre** (p. 130)'s base class.

**Returns**

Returns "gazebo::ogredynamicslines"

#### 10.59.3.6 const math::Vector3& gazebo::rendering::DynamicLines::GetPoint ( unsigned int _index ) const

Return the location of an existing point in the point list.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Number of the point to return |

**Returns**

**math::Vector3** (p. 1091) value of the point

#### 10.59.3.7 unsigned int gazebo::rendering::DynamicLines::GetPointCount ( ) const

Return the total number of points in the point list.

**Returns**

Number of points

#### 10.59.3.8 void gazebo::rendering::DynamicLines::SetColor ( unsigned int _index, const common::Color & _color )

Change the color of an existing point in the point list.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the point to set |
| in | *_color* | **common::Color** (p. 233) Pixelcolor color to set the point to |

#### 10.59.3.9 void gazebo::rendering::DynamicLines::SetPoint ( unsigned int _index, const math::Vector3 & _value )

Change the location of an existing point in the point list.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the point to set |
| in | *_value* | **math::Vector3** (p. 1091) value to set the point to |

#### 10.59.3.10 void gazebo::rendering::DynamicLines::Update ( )

Call this to update the hardware buffer after making changes.

The documentation for this class was generated from the following file:

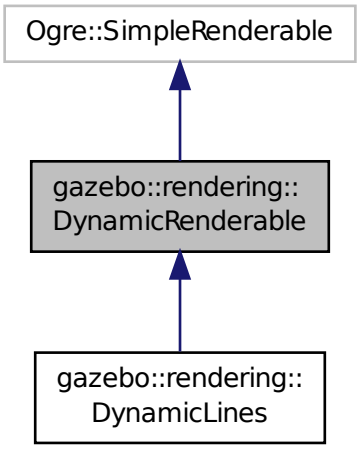
- **DynamicLines.hh**

## 10.60 gazebo::rendering::DynamicRenderable Class Reference

Abstract base class providing mechanisms for dynamically growing hardware buffers.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DynamicRenderable:




### Public Member Functions

- **DynamicRenderable** ()

  *Constructor.*
- virtual ~**DynamicRenderable** ()

  *Virtual destructor.*
- virtual Ogre::Real **getBoundingRadius** () const

  *Implementation of Ogre::SimpleRenderable.*
- std::string **GetMovableType** () const

  *Get type of movable.*
- **RenderOpType GetOperationType** () const

  *Get the render operation type.*
- virtual Ogre::Real **getSquaredViewDepth** (const Ogre::Camera *_cam) const

  *Implementation of Ogre::SimpleRenderable.*
- void **Init** (**RenderOpType** _opType, bool _useIndices=false)

  *Initializes the dynamic renderable.*
- void **SetOperationType** (**RenderOpType** _opType)

  *Set the render operation type.*

### Protected Member Functions

- virtual void **CreateVertexDeclaration** ()=0

  *Creates the vertex declaration.*
- virtual void **FillHardwareBuffers** ()=0

  *Fills the hardware vertex and index buffers with data.*
- void **PrepareHardwareBuffers** (size_t _vertexCount, size_t _indexCount)

  *Prepares the hardware buffers for the requested vertex and index counts.*

### Protected Attributes

- size_t **indexBufferCapacity**

  *Maximum capacity of the currently allocated index buffer.*
- size_t **vertexBufferCapacity**

  *Maximum capacity of the currently allocated vertex buffer.*

## 10.60.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

## 10.60.2 Constructor & Destructor Documentation

#### 10.60.2.1 gazebo::rendering::DynamicRenderable::DynamicRenderable ( )

Constructor.

#### 10.60.2.2 virtual gazebo::rendering::DynamicRenderable::~DynamicRenderable ( ) [virtual]

Virtual destructor.

## 10.60.3 Member Function Documentation

#### 10.60.3.1 virtual void gazebo::rendering::DynamicRenderable::CreateVertexDeclaration ( ) [protected], [pure virtual]

Creates the vertex declaration.

**Remarks**

Override and set mRenderOp.vertexData->vertexDeclaration here. mRenderOp.vertexData will be created for you before this method is called.

#### 10.60.3.2 virtual void gazebo::rendering::DynamicRenderable::FillHardwareBuffers ( ) [protected], [pure virtual]

Fills the hardware vertex and index buffers with data.

**Remarks**

This function must call prepareHardwareBuffers() before locking the buffers to ensure the they are large enough for the data to be written. Afterwards the vertex and index buffers (if using indices) can be locked, and data can be written to them.

#### 10.60.3.3 virtual Ogre::Real gazebo::rendering::DynamicRenderable::getBoundingRadius ( ) const [virtual]

Implementation of Ogre::SimpleRenderable.

**Returns**

The bounding radius

#### 10.60.3.4 std::string gazebo::rendering::DynamicRenderable::GetMovableType ( ) const

Get type of movable.

**Returns**

This returns "gazebo::DynamicRenderable"

#### 10.60.3.5 RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType ( ) const

Get the render operation type.

**Returns**

The render operation type.

#### 10.60.3.6 virtual Ogre::Real gazebo::rendering::DynamicRenderable::getSquaredViewDepth ( const Ogre::Camera * *_cam* ) const [virtual]

Implementation of Ogre::SimpleRenderable.

**Parameters**

| | | |
|---|---|---|
| in | *_cam* | Pointer to the **Ogre** (p. 130) camera that views the renderable. |

**Returns**

The squared depth in the **Camera** (p. 186)'s view

#### 10.60.3.7 void gazebo::rendering::DynamicRenderable::Init ( RenderOpType *_opType*, bool *_useIndices* = false )

Initializes the dynamic renderable.

**Remarks**

This function should only be called once. It initializes the render operation, and calls the abstract function **CreateVertexDeclaration()** (p. 376).

**Parameters**

| | | |
|---|---|---|
| in | *_opType* | The type of render operation to perform. |
| in | *_useIndices* | Specifies whether to use indices to determine the vertices to use as input. |

#### 10.60.3.8 void gazebo::rendering::DynamicRenderable::PrepareHardwareBuffers ( size_t *_vertexCount*, size_t *_indexCount* ) [protected]

Prepares the hardware buffers for the requested vertex and index counts.

**Remarks**

This function must be called before locking the buffers in fillHardwareBuffers(). It guarantees that the hardware buffers are large enough to hold at least the requested number of vertices and indices (if using indices). The buffers are possibly reallocated to achieve this.

**The vertex and index count in the render operation are set to**

the values of vertexCount and indexCount respectively.

**Parameters**

| | | |
|---|---|---|
| in | *_vertexCount* | The number of vertices the buffer must hold. |
| in | *_indexCount* | The number of indices the buffer must hold. This parameter is ignored if not using indices. |

#### 10.60.3.9 void gazebo::rendering::DynamicRenderable::SetOperationType ( RenderOpType *_opType* )

Set the render operation type.

**Parameters**

| | | |
|---|---|---|
| in | *_opType* | The type of render operation to perform. |

### 10.60.4 Member Data Documentation

#### 10.60.4.1 size_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

#### 10.60.4.2 size_t gazebo::rendering::DynamicRenderable::vertexBufferCapacity [protected]

Maximum capacity of the currently allocated vertex buffer.

The documentation for this class was generated from the following file:

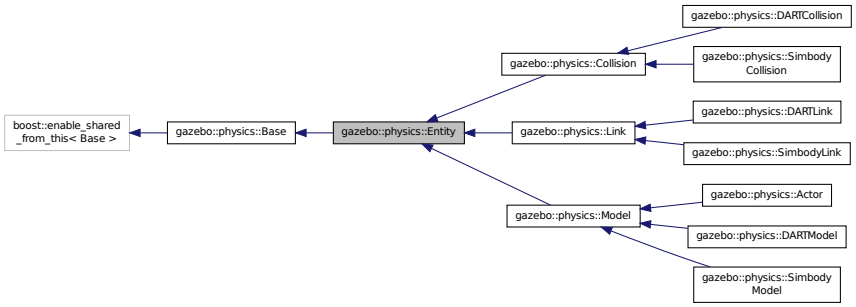
- **DynamicRenderable.hh**

## 10.61 gazebo::physics::Entity Class Reference

**Base** (p. 159) class for all physics objects in Gazebo.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Entity:




### Public Member Functions

- **Entity** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**Entity** ()

  *Destructor.*
- virtual void **Fini** ()

  *Finalize the entity.*
- virtual **math::Box GetBoundingBox** () const

  *Return the bounding box for the entity.*
- **CollisionPtr GetChildCollision** (const std::string &_name)

  *Get a child collision entity, if one exists.*
- **LinkPtr GetChildLink** (const std::string &_name)

  *Get a child linke entity, if one exists.*
- **math::Box GetCollisionBoundingBox** () const

  *Returns collision bounding box.*
- const **math::Pose** & **GetDirtyPose** () const

  *Returns **Entity::dirtyPose** (p. 389).*
- **math::Pose GetInitialRelativePose** () const

  *Get the initial relative pose.*
- void **GetNearestEntityBelow** (double &_distBelow, std::string &_entityName)

  *Get the distance to the nearest entity below (along the Z-axis) this entity.*
- **ModelPtr GetParentModel** ()

  *Get the parent model, if one exists.*
- virtual **math::Vector3 GetRelativeAngularAccel** () const

*Get the angular acceleration of the entity.*

- virtual **math::Vector3 GetRelativeAngularVel** () const

  *Get the angular velocity of the entity.*
- virtual **math::Vector3 GetRelativeLinearAccel** () const

  *Get the linear acceleration of the entity.*
- virtual **math::Vector3 GetRelativeLinearVel** () const

  *Get the linear velocity of the entity.*
- **math::Pose GetRelativePose** () const

  *Get the pose of the entity relative to its parent.*
- virtual **math::Vector3 GetWorldAngularAccel** () const

  *Get the angular acceleration of the entity in the world frame.*
- virtual **math::Vector3 GetWorldAngularVel** () const

  *Get the angular velocity of the entity in the world frame.*
- virtual **math::Vector3 GetWorldLinearAccel** () const

  *Get the linear acceleration of the entity in the world frame.*
- virtual **math::Vector3 GetWorldLinearVel** () const

  *Get the linear velocity of the entity in the world frame.*
- const **math::Pose** & **GetWorldPose** () const

  *Get the absolute pose of the entity.*
- bool **IsCanonicalLink** () const

  *A helper function that checks if this is a canonical body.*
- bool **IsStatic** () const

  *Return whether this entity is static.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the entity.*
- void **PlaceOnEntity** (const std::string &_entityName)

  *Move this entity to be ontop of another entity by name.*
- void **PlaceOnNearestEntityBelow** ()

  *Move this entity to be ontop of the nearest entity below.*
- virtual void **Reset** ()

  *Reset the entity.*
- void **SetAnimation** (const **common::PoseAnimationPtr** &_anim, boost::function< void()> _onComplete)

  *Set an animation for this entity.*
- void **SetAnimation** (**common::PoseAnimationPtr** _anim)

  *Set an animation for this entity.*
- void **SetCanonicalLink** (bool _value)

  *Set to true if this entity is a canonical link for a model.*
- void **SetInitialRelativePose** (const **math::Pose** &_pose)

  *Set the initial pose.*
- virtual void **SetName** (const std::string &_name)

  *Set the name of the entity.*
- void **SetRelativePose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)

  *Set the pose of the entity relative to its parent.*
- void **SetStatic** (const bool &_static)

  *Set whether this entity is static: immovable.*
- void **SetWorldPose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)

  *Set the world pose of the entity.*

- void **SetWorldTwist** (const **math::Vector3** &_linear, const **math::Vector3** &_angular, bool _update-Children=true)

  *Set angular and linear rates of an **physics::Entity** (p. 379).*
- virtual void **StopAnimation** ()

  *Stop the current animation, if any.*
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  *Update the parameters using new sdf values.*

### Protected Member Functions

- virtual void **OnPoseChange** ()=0

  *This function is called when the entity's (or one of its parents) pose of the parent has changed.*

### Protected Attributes

- **common::PoseAnimationPtr animation**

  *Current pose animation.*
- **event::ConnectionPtr animationConnection**

  *Connection used to update an animation.*
- **math::Pose animationStartPose**

  *Start pose of an animation.*
- std::vector< **event::ConnectionPtr** > **connections**

  *All our event connections.*
- **math::Pose dirtyPose**

  *The pose set by a physics engine.*
- **transport::NodePtr node**

  *Communication node.*
- **EntityPtr parentEntity**

  *A helper that prevents numerous dynamic_casts.*
- **common::Time prevAnimationTime**

  *Previous time an animation was updated.*
- **transport::PublisherPtr requestPub**

  *Request publisher.*
- **math::Vector3 scale**

  *Scale of the entity.*
- **transport::PublisherPtr visPub**

  *Visual publisher.*
- msgs::Visual * **visualMsg**

  *Visual message container.*

### Additional Inherited Members

### 10.61.1 Detailed Description

**Base** (p. 159) class for all physics objects in Gazebo.

### 10.61.2 Constructor & Destructor Documentation

#### 10.61.2.1 gazebo::physics::Entity::Entity ( BasePtr _parent ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the entity. |

#### 10.61.2.2 virtual gazebo::physics::Entity::~Entity ( ) [virtual]

Destructor.

### 10.61.3 Member Function Documentation

#### 10.61.3.1 virtual void gazebo::physics::Entity::Fini ( ) [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 165).

Reimplemented in **gazebo::physics::Actor** (p. 134), **gazebo::physics::Link** (p. 550), **gazebo::physics::Model** (p. 629), **gazebo::physics::DARTModel** (p. 328), **gazebo::physics::Collision** (p. 223), **gazebo::physics::DARTLink** (p. 318), **gazebo::physics::SimbodyLink** (p. 905), and **gazebo::physics::DARTCollision** (p. 288).

#### 10.61.3.2 virtual math::Box gazebo::physics::Entity::GetBoundingBox ( ) const [virtual]

Return the bounding box for the entity.

**Returns**

The bounding box.

Reimplemented in **gazebo::physics::Link** (p. 550), **gazebo::physics::Model** (p. 629), **gazebo::physics::Collision** (p. 223), **gazebo::physics::DARTCollision** (p. 288), and **gazebo::physics::SimbodyCollision** (p. 874).

#### 10.61.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision ( const std::string & _name )

Get a child collision entity, if one exists.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the child collision object. |

**Returns**

Pointer to the **Collision** (p. 220) object, or NULL if not found.

#### 10.61.3.4 LinkPtr gazebo::physics::Entity::GetChildLink ( const std::string & _name )

Get a child linke entity, if one exists.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the child **Link** (p. 542) object. |

**Returns**

Pointer to the **Link** (p. 542) object, or NULL if not found.

#### 10.61.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox ( ) const

Returns collision bounding box.

**Returns**

Collsiion boundin box.

#### 10.61.3.6 const math::Pose& gazebo::physics::Entity::GetDirtyPose ( ) const

Returns **Entity::dirtyPose** (p. 389).

The dirty pose is the pose set by the physics engine before it's value is propagated to the rest of the simulator.

**Returns**

The dirty pose of the entity.

#### 10.61.3.7 math::Pose gazebo::physics::Entity::GetInitialRelativePose ( ) const

Get the initial relative pose.

**Returns**

The initial relative pose.

#### 10.61.3.8 void gazebo::physics::Entity::GetNearestEntityBelow ( double & _distBelow, std::string & _entityName )

Get the distance to the nearest entity below (along the Z-axis) this entity.

**Parameters**

| | | |
|---|---|---|
| out | *_distBelow* | The distance to the nearest entity below. |
| out | *_entityName* | The name of the nearest entity below. |

#### 10.61.3.9 ModelPtr gazebo::physics::Entity::GetParentModel ( )

Get the parent model, if one exists.

**Returns**

Pointer to a model, or NULL if no parent model exists.

#### 10.61.3.10 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity.

**Returns**

A **math::Vector3** (p. 1091) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 553), **gazebo::physics::Collision** (p. 224), and **gazebo::physics::Model** (p. 631).

#### 10.61.3.11 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity.

**Returns**

A **math::Vector3** (p. 1091) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 553), **gazebo::physics::Collision** (p. 224), and **gazebo::physics::Model** (p. 631).

#### 10.61.3.12 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity.

**Returns**

A **math::Vector3** (p. 1091) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 553), **gazebo::physics::Collision** (p. 225), and **gazebo::physics::Model** (p. 632).

#### 10.61.3.13 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity.

**Returns**

A **math::Vector3** (p. 1091) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 553), **gazebo::physics::Collision** (p. 225), and **gazebo::physics::Model** (p. 632).

#### 10.61.3.14 math::Pose gazebo::physics::Entity::GetRelativePose ( ) const

Get the pose of the entity relative to its parent.

**Returns**

The pose of the entity relative to its parent.

#### 10.61.3.15 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity in the world frame.

**Returns**

A **math::Vector3** (p. 1091) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 554), **gazebo::physics::Collision** (p. 226), and **gazebo::physics::Model** (p. 632).

#### 10.61.3.16 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity in the world frame.

**Returns**

A **math::Vector3** (p. 1091) for the velocity.

Reimplemented in **gazebo::physics::Collision** (p. 226), **gazebo::physics::Model** (p. 633), **gazebo::physics::DARTLink** (p. 319), and **gazebo::physics::SimbodyLink** (p. 905).

#### 10.61.3.17 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity in the world frame.

**Returns**

A **math::Vector3** (p. 1091) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 555), **gazebo::physics::Collision** (p. 226), and **gazebo::physics::Model** (p. 633).

#### 10.61.3.18 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity in the world frame.

**Returns**

A **math::Vector3** (p. 1091) for the linear velocity.

Reimplemented in **gazebo::physics::Collision** (p. 226), and **gazebo::physics::Model** (p. 633).

#### 10.61.3.19 const math::Pose& gazebo::physics::Entity::GetWorldPose ( ) const [inline]

Get the absolute pose of the entity.

**Returns**

The absolute pose of the entity.

#### 10.61.3.20 bool gazebo::physics::Entity::IsCanonicalLink ( ) const [inline]

A helper function that checks if this is a canonical body.

**Returns**

True if the link is canonical.

#### 10.61.3.21 bool gazebo::physics::Entity::IsStatic ( ) const

Return whether this entity is static.

**Returns**

True if static.

#### 10.61.3.22 virtual void gazebo::physics::Entity::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the entity.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to an SDF element. |

Reimplemented from **gazebo::physics::Base** (p. 168).

Reimplemented in **gazebo::physics::Link** (p. 556), **gazebo::physics::Actor** (p. 134), **gazebo::physics::Model** (p. 633), **gazebo::physics::Collision** (p. 227), **gazebo::physics::SimbodyCollision** (p. 874), **gazebo::physics::DARTLink** (p. 321), **gazebo::physics::SimbodyLink** (p. 907), **gazebo::physics::DARTModel** (p. 328), **gazebo::physics::DARTCollision** (p. 289), and **gazebo::physics::SimbodyModel** (p. 913).

#### 10.61.3.23 virtual void gazebo::physics::Entity::OnPoseChange ( ) [protected], [pure virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implemented in **gazebo::physics::Link** (p. 557), **gazebo::physics::Model** (p. 634), **gazebo::physics::DARTLink** (p. 321), **gazebo::physics::SimbodyLink** (p. 907), **gazebo::physics::DARTCollision** (p. 290), and **gazebo::physics::SimbodyCollision** (p. 875).

#### 10.61.3.24 void gazebo::physics::Entity::PlaceOnEntity ( const std::string & *_entityName* )

Move this entity to be ontop of another entity by name.

**Parameters**

| | | |
|---|---|---|
| in | *_entityName* | Name of the **Entity** (p. 379) this **Entity** (p. 379) should be ontop of. |

#### 10.61.3.25 void gazebo::physics::Entity::PlaceOnNearestEntityBelow ( )

Move this entity to be ontop of the nearest entity below.

#### 10.61.3.26 virtual void gazebo::physics::Entity::Reset ( ) [virtual]

Reset the entity.

Reimplemented from **gazebo::physics::Base** (p. 169).

Reimplemented in **gazebo::physics::Model** (p. 634), and **gazebo::physics::Link** (p. 557).

#### 10.61.3.27 void gazebo::physics::Entity::SetAnimation ( const common::PoseAnimationPtr & *_anim,* boost::function< void() > *_onComplete* )

Set an animation for this entity.

**Parameters**

| | | |
|---|---|---|
| in | *_anim* | Pose animation. |
| in | *_onComplete* | Callback for when the animation completes. |

#### 10.61.3.28 void gazebo::physics::Entity::SetAnimation ( common::PoseAnimationPtr *_anim* )

Set an animation for this entity.

**Parameters**

| | | |
|---|---|---|
| in | *_anim* | Pose animation. |

#### 10.61.3.29 void gazebo::physics::Entity::SetCanonicalLink ( bool *_value* )

Set to true if this entity is a canonical link for a model.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | True if the link is canonical. |

#### 10.61.3.30 void gazebo::physics::Entity::SetInitialRelativePose ( const math::Pose & *_pose* )

Set the initial pose.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | The initial pose. |

#### 10.61.3.31 virtual void gazebo::physics::Entity::SetName ( const std::string & _name ) [virtual]

Set the name of the entity.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The new name. |

Reimplemented from **gazebo::physics::Base** (p. 170).

#### 10.61.3.32 void gazebo::physics::Entity::SetRelativePose ( const math::Pose & _pose, bool _notify = true, bool _publish = true )

Set the pose of the entity relative to its parent.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | The new pose. |
| in | *_notify* | True = tell children of the pose change. |
| in | *_publish* | True to publish the pose. |

#### 10.61.3.33 void gazebo::physics::Entity::SetStatic ( const bool & _static )

Set whether this entity is static: immovable.

**Parameters**

| | | |
|---|---|---|
| in | *_static* | True = static. |

#### 10.61.3.34 void gazebo::physics::Entity::SetWorldPose ( const math::Pose & _pose, bool _notify = true, bool _publish = true )

Set the world pose of the entity.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | The new world pose. |
| in | *_notify* | True = tell children of the pose change. |
| in | *_publish* | True to publish the pose. |

#### 10.61.3.35 void gazebo::physics::Entity::SetWorldTwist ( const math::Vector3 & _linear, const math::Vector3 & _angular, bool _updateChildren = true )

Set angular and linear rates of an **physics::Entity** (p. 379).

**Parameters**

| | | |
|---|---|---|
| in | *_linear* | Linear twist. |
| in | *_angular* | Angular twist. |
| in | *_updateChildren* | True to pass this update to child entities. |

#### 10.61.3.36 virtual void gazebo::physics::Entity::StopAnimation ( ) [virtual]

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 637).

#### 10.61.3.37 virtual void gazebo::physics::Entity::UpdateParameters ( sdf::ElementPtr _sdf ) [virtual]

Update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF to update from. |

Reimplemented from **gazebo::physics::Base** (p. 171).

Reimplemented in **gazebo::physics::Actor** (p. 135), **gazebo::physics::Link** (p. 562), **gazebo::physics::Model** (p. 638), and **gazebo::physics::Collision** (p. 229).

### 10.61.4 Member Data Documentation

#### 10.61.4.1 common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]

Current pose animation.

#### 10.61.4.2 event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]

Connection used to update an animation.

#### 10.61.4.3 math::Pose gazebo::physics::Entity::animationStartPose [protected]

Start pose of an animation.

#### 10.61.4.4 std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections [protected]

All our event connections.

#### 10.61.4.5 math::Pose gazebo::physics::Entity::dirtyPose [protected]

The pose set by a physics engine.

#### 10.61.4.6 transport::NodePtr gazebo::physics::Entity::node [protected]

Communication node.

#### 10.61.4.7 EntityPtr gazebo::physics::Entity::parentEntity [protected]

A helper that prevents numerous dynamic_casts.

#### 10.61.4.8 common::Time gazebo::physics::Entity::prevAnimationTime [protected]

Previous time an animation was updated.

#### 10.61.4.9 transport::PublisherPtr gazebo::physics::Entity::requestPub [protected]

Request publisher.

#### 10.61.4.10 math::Vector3 gazebo::physics::Entity::scale [protected]

Scale of the entity.

#### 10.61.4.11 transport::PublisherPtr gazebo::physics::Entity::visPub [protected]

Visual publisher.

#### 10.61.4.12 msgs::Visual* gazebo::physics::Entity::visualMsg [protected]

Visual message container.

The documentation for this class was generated from the following file:

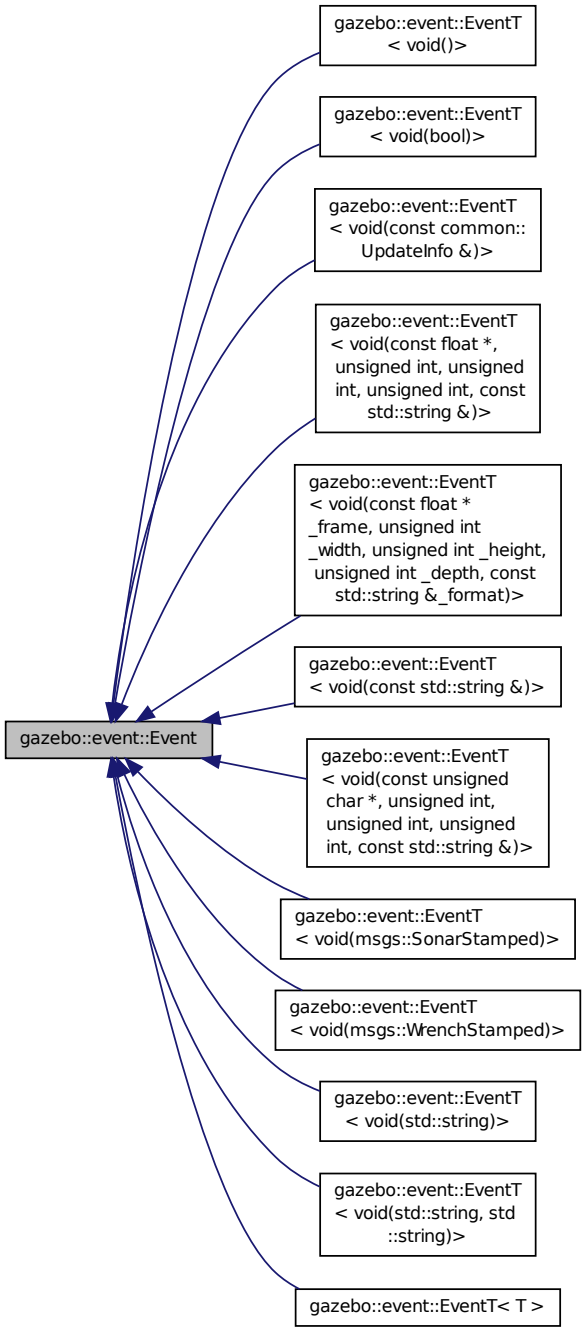
- **Entity.hh**

## 10.62 gazebo::event::Event Class Reference

Base class for all events.

`#include <common/common.hh>`

Inheritance diagram for gazebo::event::Event:




## Public Member Functions

- virtual ~**Event** ()

*Constructor.*

- virtual void **Disconnect** (**ConnectionPtr** _c)=0

  *Disconnect.*

- virtual void **Disconnect** (int _id)=0

  *Disconnect.*

### 10.62.1 Detailed Description

Base class for all events.

### 10.62.2 Constructor & Destructor Documentation

#### 10.62.2.1 virtual gazebo::event::Event::~Event ( ) [inline], [virtual]

Constructor.

### 10.62.3 Member Function Documentation

#### 10.62.3.1 virtual void gazebo::event::Event::Disconnect ( ConnectionPtr *_c* ) [pure virtual]

Disconnect.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | A pointer to a connection |

Implemented in **gazebo::event::EventT< T >** (p. 46), **gazebo::event::EventT< void(msgs::SonarStamped)>** (p. 46), **gazebo::event::EventT< void(std::string)>** (p. 46), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 46), **gazebo::event::EventT< void(msgs::WrenchStamped)>** (p. 46), **gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format)>** (p. 46), **gazebo::event::EventT< void(const std::string &)>** (p. 46), **gazebo::event::EventT< void()>** (p. 46), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 46), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 46), **gazebo::event::EventT< void(std::string, std::string)>** (p. 46), and **gazebo::event::EventT< void(bool)>** (p. 46).

#### 10.62.3.2 virtual void gazebo::event::Event::Disconnect ( int *_id* ) [pure virtual]

Disconnect.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | Integer ID of a connection |

Implemented in **gazebo::event::EventT< T >** (p. 47), **gazebo::event::EventT< void(msgs::SonarStamped)>** (p. 47), **gazebo::event::EventT< void(std::string)>** (p. 47), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 47), **gazebo::event::EventT< void(msgs::WrenchStamped)>** (p. 47), **gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format)>** (p. 47), **gazebo::event::EventT< void(const std::string &)>** (p. 47), **gazebo::event::EventT< void()>** (p. 47), **gazebo::event::EventT< void(const common::-**

**UpdateInfo &)>** (p. 47), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 47), **gazebo::event::EventT< void(std::string, std::string)>** (p. 47), and **gazebo::event::EventT< void(bool)>** (p. 47).

The documentation for this class was generated from the following file:

- **Event.hh**

## 10.63 gazebo::event::Events Class Reference

An **Event** (p. 390) class to get notifications for simulator events.

`#include <common/common.hh>`

### Static Public Member Functions

- template<typename T >
  static **ConnectionPtr ConnectAddEntity** (T _subscriber)

  *Connect a boost::slot the the add entity signal.*
- template<typename T >
  static **ConnectionPtr ConnectCreateEntity** (T _subscriber)

  *Connect a boost::slot the the add entity signal.*
- template<typename T >
  static **ConnectionPtr ConnectDeleteEntity** (T _subscriber)

  *Connect a boost::slot the delete entity.*
- template<typename T >
  static **ConnectionPtr ConnectDiagTimerStart** (T _subscriber)

  *Connect a boost::slot the diagnostic timer start signal.*
- template<typename T >
  static **ConnectionPtr ConnectDiagTimerStop** (T _subscriber)

  *Connect a boost::slot the diagnostic timer stop signal.*
- template<typename T >
  static **ConnectionPtr ConnectPause** (T _subscriber)

  *Connect a boost::slot the the pause signal.*
- template<typename T >
  static **ConnectionPtr ConnectPostRender** (T _subscriber)

  *Connect a boost::slot the post render update signal.*
- template<typename T >
  static **ConnectionPtr ConnectPreRender** (T _subscriber)

  *Render start signal.*
- template<typename T >
  static **ConnectionPtr ConnectRender** (T _subscriber)

  *Connect a boost::slot the render update signal.*
- template<typename T >
  static **ConnectionPtr ConnectSetSelectedEntity** (T _subscriber)

  *Connect a boost::slot the set selected entity.*
- template<typename T >
  static **ConnectionPtr ConnectSigInt** (T _subscriber)

  *Connect a boost::slot to the sigint event.*

- template<typename T >
  static **ConnectionPtr ConnectStep** (T _subscriber)

  *Connect a boost::slot the the step signal.*
- template<typename T >
  static **ConnectionPtr ConnectStop** (T _subscriber)

  *Connect a boost::slot the the stop signal.*
- template<typename T >
  static **ConnectionPtr ConnectWorldCreated** (T _subscriber)

  *Connect a boost::slot the the world created signal.*
- template<typename T >
  static **ConnectionPtr ConnectWorldUpdateBegin** (T _subscriber)

  *Connect a boost::slot the the world update start signal.*
- template<typename T >
  static **ConnectionPtr ConnectWorldUpdateEnd** (T _subscriber)

  *Connect a boost::slot the the world update end signal.*
- static void **DisconnectAddEntity** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the add entity signal.*
- static void **DisconnectCreateEntity** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the add entity signal.*
- static void **DisconnectDeleteEntity** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the delete entity.*
- static void **DisconnectDiagTimerStart** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the diagnostic timer start signal.*
- static void **DisconnectDiagTimerStop** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the diagnostic timer stop signal.*
- static void **DisconnectPause** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the pause signal.*
- static void **DisconnectPostRender** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the post render update signal.*
- static void **DisconnectPreRender** (**ConnectionPtr** _subscriber)

  *Disconnect a render start signal.*
- static void **DisconnectRender** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the render update signal.*
- static void **DisconnectSetSelectedEntity** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the set selected entity.*
- static void **DisconnectSigInt** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot to the sigint event.*
- static void **DisconnectStep** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the step signal.*
- static void **DisconnectStop** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the stop signal.*
- static void **DisconnectWorldCreated** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the world created signal.*
- static void **DisconnectWorldUpdateBegin** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the world update start signal.*
- static void **DisconnectWorldUpdateEnd** (**ConnectionPtr** _subscriber)

  *Disconnect a boost::slot the the world update end signal.*

### Static Public Attributes

- static **EventT**< void(std::string)> **addEntity**

  *An entity has been added.*
- static **EventT**< void(std::string)> **deleteEntity**

  *An entity has been deleted.*
- static **EventT**< void(std::string)> **diagTimerStart**

  *Diagnostic timer start.*
- static **EventT**< void(std::string)> **diagTimerStop**

  *Diagnostic timer stop.*
- static **EventT**< void(std::string)> **entityCreated**

  *An entity has been created.*
- static **EventT**< void(bool)> **pause**

  *Pause signal.*
- static **EventT**< void()> **postRender**

  *Post-Render.*
- static **EventT**< void()> **preRender**

  *Pre-render.*
- static **EventT**< void()> **render**

  *Render.*
- static **EventT**< void(std::string, std::string)> **setSelectedEntity**

  *An entity has been selected.*
- static **EventT**< void()> **sigInt**

  *Simulation stop signal.*
- static **EventT**< void()> **step**

  *Step the simulation once signal.*
- static **EventT**< void()> **stop**

  *Simulation stop signal.*
- static **EventT**< void(std::string)> **worldCreated**

  *A world has been created.*
- static **EventT**< void(const **common::UpdateInfo** &)> **worldUpdateBegin**

  *World update has started.*
- static **EventT**< void()> **worldUpdateEnd**

  *World update has ended.*

## 10.63.1 Detailed Description

An **Event** (p. 390) class to get notifications for simulator events.

## 10.63.2 Member Function Documentation

### 10.63.2.1 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectAddEntity ( T *_subscriber* ) [inline], [static]

Connect a boost::slot the the add entity signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References addEntity, and gazebo::event::EventT< T >::Connect().

#### 10.63.2.2 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectCreateEntity ( T *_subscriber* ) [inline], [static]

Connect a boost::slot the the add entity signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and entityCreated.

#### 10.63.2.3 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDeleteEntity ( T *_subscriber* ) [inline], [static]

Connect a boost::slot the delete entity.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and deleteEntity.

#### 10.63.2.4 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStart ( T *_subscriber* ) [inline], [static]

Connect a boost::slot the diagnostic timer start signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and diagTimerStart.

#### 10.63.2.5 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop ( T _subscriber ) [inline], [static]

Connect a boost::slot the diagnostic timer stop signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and diagTimerStop.

#### 10.63.2.6 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPause ( T _subscriber ) [inline], [static]

Connect a boost::slot the the pause signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and pause.

#### 10.63.2.7 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPostRender ( T _subscriber ) [inline], [static]

Connect a boost::slot the post render update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and postRender.

#### 10.63.2.8 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPreRender ( T _subscriber ) [inline], [static]

Render start signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and preRender.

#### 10.63.2.9 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectRender ( T *_subscriber* ) [inline], [static]

Connect a boost::slot the render update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and render.

#### 10.63.2.10 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity ( T *_subscriber* ) [inline], [static]

Connect a boost::slot the set selected entity.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and setSelectedEntity.

#### 10.63.2.11 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSigInt ( T *_subscriber* ) [inline], [static]

Connect a boost::slot to the sigint event.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and sigInt.

#### 10.63.2.12 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep ( T _subscriber ) [inline], [static]

Connect a boost::slot the the step signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and step.

#### 10.63.2.13 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStop ( T _subscriber ) [inline], [static]

Connect a boost::slot the the stop signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and stop.

#### 10.63.2.14 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldCreated ( T _subscriber ) [inline], [static]

Connect a boost::slot the the world created signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and worldCreated.

#### 10.63.2.15 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateBegin ( T _subscriber ) [inline], [static]

Connect a boost::slot the the world update start signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and worldUpdateBegin.

#### 10.63.2.16 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateEnd ( T *_subscriber* ) [inline], [static]

Connect a boost::slot the the world update end signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

**Returns**

a connection

References gazebo::event::EventT< T >::Connect(), and worldUpdateEnd.

#### 10.63.2.17 static void gazebo::event::Events::DisconnectAddEntity ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the the add entity signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References addEntity, and gazebo::event::EventT< T >::Disconnect().

#### 10.63.2.18 static void gazebo::event::Events::DisconnectCreateEntity ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the the add entity signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and entityCreated.

#### 10.63.2.19 static void gazebo::event::Events::DisconnectDeleteEntity ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the delete entity.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References deleteEntity, and gazebo::event::EventT< T >::Disconnect().

#### 10.63.2.20 static void gazebo::event::Events::DisconnectDiagTimerStart ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the diagnostic timer start signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References diagTimerStart, and gazebo::event::EventT< T >::Disconnect().

#### 10.63.2.21 static void gazebo::event::Events::DisconnectDiagTimerStop ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the diagnostic timer stop signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References diagTimerStop, and gazebo::event::EventT< T >::Disconnect().

#### 10.63.2.22 static void gazebo::event::Events::DisconnectPause ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the the pause signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and pause.

#### 10.63.2.23 static void gazebo::event::Events::DisconnectPostRender ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the post render update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and postRender.

#### 10.63.2.24 static void gazebo::event::Events::DisconnectPreRender ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a render start signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and preRender.

#### 10.63.2.25 static void gazebo::event::Events::DisconnectRender ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the render update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and render.

#### 10.63.2.26 static void gazebo::event::Events::DisconnectSetSelectedEntity ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the set selected entity.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and setSelectedEntity.

#### 10.63.2.27 static void gazebo::event::Events::DisconnectSigInt ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot to the sigint event.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and sigInt.

#### 10.63.2.28 static void gazebo::event::Events::DisconnectStep ( ConnectionPtr *_subscriber* ) [inline], [static]

Disconnect a boost::slot the the step signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and step.

#### 10.63.2.29 static void gazebo::event::Events::DisconnectStop ( ConnectionPtr _subscriber ) [inline], [static]

Disconnect a boost::slot the the stop signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and stop.

#### 10.63.2.30 static void gazebo::event::Events::DisconnectWorldCreated ( ConnectionPtr _subscriber ) [inline], [static]

Disconnect a boost::slot the the world created signal.

References gazebo::event::EventT< T >::Disconnect(), and worldCreated.

#### 10.63.2.31 static void gazebo::event::Events::DisconnectWorldUpdateBegin ( ConnectionPtr _subscriber ) [static]

Disconnect a boost::slot the the world update start signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

#### 10.63.2.32 static void gazebo::event::Events::DisconnectWorldUpdateEnd ( ConnectionPtr _subscriber ) [inline], [static]

Disconnect a boost::slot the the world update end signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | the subscriber to this event |

References gazebo::event::EventT< T >::Disconnect(), and worldUpdateEnd.

### 10.63.3 Member Data Documentation

#### 10.63.3.1 EventT<void (std::string)> gazebo::event::Events::addEntity [static]

An entity has been added.

Referenced by ConnectAddEntity(), and DisconnectAddEntity().

#### 10.63.3.2 EventT<void (std::string)> gazebo::event::Events::deleteEntity [static]

An entity has been deleted.

Referenced by ConnectDeleteEntity(), and DisconnectDeleteEntity().

#### 10.63.3.3 EventT<void (std::string)> gazebo::event::Events::diagTimerStart [static]

Diagnostic timer start.

Referenced by ConnectDiagTimerStart(), and DisconnectDiagTimerStart().

#### 10.63.3.4 EventT<void (std::string)> gazebo::event::Events::diagTimerStop [static]

Diagnostic timer stop.

Referenced by ConnectDiagTimerStop(), and DisconnectDiagTimerStop().

#### 10.63.3.5 EventT<void (std::string)> gazebo::event::Events::entityCreated [static]

An entity has been created.

Referenced by ConnectCreateEntity(), and DisconnectCreateEntity().

#### 10.63.3.6 EventT<void (bool)> gazebo::event::Events::pause [static]

Pause signal.

Referenced by ConnectPause(), and DisconnectPause().

#### 10.63.3.7 EventT<void ()> gazebo::event::Events::postRender [static]

Post-Render.

Referenced by ConnectPostRender(), and DisconnectPostRender().

#### 10.63.3.8 EventT<void ()> gazebo::event::Events::preRender [static]

Pre-render.

Referenced by ConnectPreRender(), and DisconnectPreRender().

#### 10.63.3.9 EventT<void ()> gazebo::event::Events::render [static]

Render.

Referenced by ConnectRender(), and DisconnectRender().

#### 10.63.3.10 EventT<void (std::string, std::string)> gazebo::event::Events::setSelectedEntity [static]

An entity has been selected.

Referenced by ConnectSetSelectedEntity(), and DisconnectSetSelectedEntity().

#### 10.63.3.11 EventT<void ()> gazebo::event::Events::sigInt [static]

Simulation stop signal.

Referenced by ConnectSigInt(), and DisconnectSigInt().

#### 10.63.3.12 EventT<void ()> gazebo::event::Events::step [static]

Step the simulation once signal.

Referenced by ConnectStep(), and DisconnectStep().

#### 10.63.3.13 EventT<void ()> gazebo::event::Events::stop [static]

Simulation stop signal.

Referenced by ConnectStop(), and DisconnectStop().

#### 10.63.3.14 EventT<void (std::string)> gazebo::event::Events::worldCreated [static]

A world has been created.

Referenced by ConnectWorldCreated(), and DisconnectWorldCreated().

#### 10.63.3.15 EventT<void (const common::UpdateInfo &)> gazebo::event::Events::worldUpdateBegin [static]

World update has started.

Referenced by ConnectWorldUpdateBegin().

#### 10.63.3.16 EventT<void ()> gazebo::event::Events::worldUpdateEnd [static]

World update has ended.

Referenced by ConnectWorldUpdateEnd(), and DisconnectWorldUpdateEnd().

The documentation for this class was generated from the following file:

- **Events.hh**

## 10.64 gazebo::rendering::Events Class Reference

Base class for rendering events.

`#include <rendering/rendering.hh>`

### Static Public Member Functions

- template<typename T >
  static **event::ConnectionPtr ConnectCreateScene** (T _subscriber)
  *Connect to a scene created event.*
- template<typename T >
  static **event::ConnectionPtr ConnectRemoveScene** (T _subscriber)
  *Connect to a scene removed event.*
- static void **DisconnectCreateScene** (**event::ConnectionPtr** _connection)
  *Disconnect from a scene created event.*
- static void **DisconnectRemoveScene** (**event::ConnectionPtr** _connection)
  *Disconnect from a scene removed event.*

### Static Public Attributes

- static **event::EventT**< void(const std::string &)> **createScene**

  *The event used to trigger a create scene event.*
- static **event::EventT**< void(const std::string &)> **removeScene**

  *The event used to trigger a remve scene event.*

## 10.64.1 Detailed Description

Base class for rendering events.

## 10.64.2 Member Function Documentation

### 10.64.2.1 template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene ( T _subscriber ) [inline], [static]

Connect to a scene created event.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback to trigger when event occurs. |

**Returns**

Pointer the connection. This must stay in scope.

References gazebo::event::EventT< T >::Connect(), and createScene.

### 10.64.2.2 template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene ( T _subscriber ) [inline], [static]

Connect to a scene removed event.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback to trigger when event occurs. |

**Returns**

Pointer the connection. This must stay in scope.

References gazebo::event::EventT< T >::Connect(), and removeScene.

### 10.64.2.3 static void gazebo::rendering::Events::DisconnectCreateScene ( event::ConnectionPtr _connection ) [inline], [static]

Disconnect from a scene created event.

**Parameters**

| | | |
|---|---|---|
| in | *_connection* | The connection to disconnect. |

References createScene, and gazebo::event::EventT< T >::Disconnect().

#### 10.64.2.4 static void gazebo::rendering::Events::DisconnectRemoveScene ( event::ConnectionPtr *_connection* ) [inline], [static]

Disconnect from a scene removed event.

**Parameters**

| | | |
|---|---|---|
| in | *_connection* | The connection to disconnect. |

References gazebo::event::EventT< T >::Disconnect(), and removeScene.

### 10.64.3 Member Data Documentation

#### 10.64.3.1 event::EventT<void (const std::string &)> gazebo::rendering::Events::createScene [static]

The event used to trigger a create scene event.

Referenced by ConnectCreateScene(), and DisconnectCreateScene().

#### 10.64.3.2 event::EventT<void (const std::string &)> gazebo::rendering::Events::removeScene [static]

The event used to trigger a remve scene event.

Referenced by ConnectRemoveScene(), and DisconnectRemoveScene().

The documentation for this class was generated from the following file:

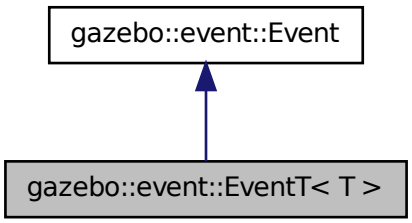
- **RenderEvents.hh**

## 10.65 gazebo::event::EventT< T > Class Template Reference

A class for event processing.

`#include <common/common.hh>`

Inheritance diagram for gazebo::event::EventT< T >:




## Public Member Functions

- virtual ~**EventT** ()

  *Destructor.*
- **ConnectionPtr Connect** (const boost::function< T > &_subscriber)

  *Connect a callback to this event.*
- unsigned int **ConnectionCount** () const

  *Get the number of connections.*
- virtual void **Disconnect** (**ConnectionPtr** _c)

  *Disconnect a callback to this event.*
- virtual void **Disconnect** (int _id)

  *Disconnect a callback to this event.*
- void **operator()** ()

  *Access the signal.*
- template<typename P >
  void **operator()** (const P &_p)

  *Signal the event with one parameter.*
- template<typename P1 , typename P2 >
  void **operator()** (const P1 &_p1, const P2 &_p2)

  *Signal the event with two parameters.*
- template<typename P1 , typename P2 , typename P3 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3)

  *Signal the event with three parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)

  *Signal the event with four parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)

  *Signal the event with five parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)

  *Signal the event with six parameters.*

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)

  *Signal the event with seven parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)

  *Signal the event with eight parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)

  *Signal the event with nine parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)

  *Signal the event with ten parameters.*
- void **Signal** ()

  *Signal the event for all subscribers.*
- template<typename P >
  void **Signal** (const P &_p)

  *Signal the event with one parameter.*
- template<typename P1 , typename P2 >
  void **Signal** (const P1 &_p1, const P2 &_p2)

  *Signal the event with two parameter.*
- template<typename P1 , typename P2 , typename P3 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3)

  *Signal the event with three parameter.*
- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)

  *Signal the event with four parameter.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)

  *Signal the event with five parameter.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)

  *Signal the event with six parameter.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)

  *Signal the event with seven parameter.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)

  *Signal the event with eight parameter.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)

*Signal the event with nine parameter.*

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)

  *Signal the event with ten parameter.*

## 10.65.1 Detailed Description

**template<typename T>class gazebo::event::EventT< T >**

A class for event processing.

## 10.65.2 Member Function Documentation

### 10.65.2.1 template<typename T> void gazebo::event::EventT< T >::operator() ( ) [inline]

Access the signal.

### 10.65.2.2 template<typename T> template<typename P > void gazebo::event::EventT< T >::operator() ( const P & _p ) [inline]

Signal the event with one parameter.

**Parameters**

| | | |
|---|---|---|
| in | *_p* | the parameter |

### 10.65.2.3 template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::operator() ( const P1 & *_p1*, const P2 & *_p2* ) [inline]

Signal the event with two parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |

### 10.65.2.4 template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::operator() ( const P1 & *_p1*, const P2 & *_p2*, const P3 & *_p3* ) [inline]

Signal the event with three parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |

#### 10.65.2.5 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4 ) [inline]

Signal the event with four parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |

#### 10.65.2.6 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5 ) [inline]

Signal the event with five parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fift parameter |

#### 10.65.2.7 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6 ) [inline]

Signal the event with six parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fift parameter |
| in | *_p6* | the sixt parameter |

#### 10.65.2.8 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7 ) [inline]

Signal the event with seven parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fifth parameter |
| in | *_p6* | the sixth parameter |
| in | *_p7* | the seventh parameter |

#### 10.65.2.9 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::operator() ( const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8* ) [inline]

Signal the event with eight parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fifth parameter |
| in | *_p6* | the sixth parameter |
| in | *_p7* | the seventh parameter |
| in | *_p8* | the eighth parameter |

#### 10.65.2.10 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::operator() ( const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8,* const P9 & *_p9* ) [inline]

Signal the event with nine parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fifth parameter |
| in | *_p6* | the sixth parameter |
| in | *_p7* | the seventh parameter |
| in | *_p8* | the eighth parameter |
| in | *_p9* | the ninth parameter |

#### 10.65.2.11 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10 ) [inline]

Signal the event with ten parameters.

**Parameters**

| | | |
|---|---|---|
| in | _p1 | the first parameter |
| in | _p2 | the second parameter |
| in | _p3 | the second parameter |
| in | _p4 | the first parameter |
| in | _p5 | the fifth parameter |
| in | _p6 | the sixth parameter |
| in | _p7 | the seventh parameter |
| in | _p8 | the eighth parameter |
| in | _p9 | the ninth parameter |
| in | _p10 | the tenth parameter |

#### 10.65.2.12 template<typename T> void gazebo::event::EventT< T >::Signal ( ) [inline]

Signal the event for all subscribers.

Referenced by gazebo::event::EventT< void(bool)>::operator()().

#### 10.65.2.13 template<typename T> template<typename P > void gazebo::event::EventT< T >::Signal ( const P & _p ) [inline]

Signal the event with one parameter.

**Parameters**

| | | |
|---|---|---|
| in | _p | parameter |

#### 10.65.2.14 template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2 ) [inline]

Signal the event with two parameter.

**Parameters**

| | | |
|---|---|---|
| in | _p1 | the first parameter |
| in | _p2 | the second parameter |

#### 10.65.2.15 template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3 ) [inline]

Signal the event with three parameter.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |

#### 10.65.2.16 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::Signal ( const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4* ) [inline]

Signal the event with four parameter.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |

#### 10.65.2.17 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::Signal ( const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5* ) [inline]

Signal the event with five parameter.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fifth parameter |

#### 10.65.2.18 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::Signal ( const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6* ) [inline]

Signal the event with six parameter.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fifth parameter |
| in | *_p6* | the sixth parameter |

#### 10.65.2.19 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7 ) [inline]

Signal the event with seven parameter.

**Parameters**

| | | |
|---|---|---|
| in | _p1 | the first parameter |
| in | _p2 | the second parameter |
| in | _p3 | the second parameter |
| in | _p4 | the first parameter |
| in | _p5 | the fifth parameter |
| in | _p6 | the sixth parameter |
| in | _p7 | the seventh parameter |

#### 10.65.2.20 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8 ) [inline]

Signal the event with eight parameter.

**Parameters**

| | | |
|---|---|---|
| in | _p1 | the first parameter |
| in | _p2 | the second parameter |
| in | _p3 | the second parameter |
| in | _p4 | the first parameter |
| in | _p5 | the fifth parameter |
| in | _p6 | the sixth parameter |
| in | _p7 | the seventh parameter |
| in | _p8 | the eighth parameter |

#### 10.65.2.21 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9 ) [inline]

Signal the event with nine parameter.

**Parameters**

| | | |
|---|---|---|
| in | _p1 | the first parameter |
| in | _p2 | the second parameter |
| in | _p3 | the second parameter |
| in | _p4 | the first parameter |
| in | _p5 | the fifth parameter |
| in | _p6 | the sixth parameter |
| in | _p7 | the seventh parameter |
| in | _p8 | the eighth parameter |
| in | _p9 | the ninth parameter |

#### 10.65.2.22 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10 ) [inline]

Signal the event with ten parameter.

**Parameters**

| | | |
|---|---|---|
| in | *_p1* | the first parameter |
| in | *_p2* | the second parameter |
| in | *_p3* | the second parameter |
| in | *_p4* | the first parameter |
| in | *_p5* | the fifth parameter |
| in | *_p6* | the sixth parameter |
| in | *_p7* | the seventh parameter |
| in | *_p8* | the eighth parameter |
| in | *_p9* | the ninth parameter |
| in | *_p10* | the tenth parameter |

The documentation for this class was generated from the following file:

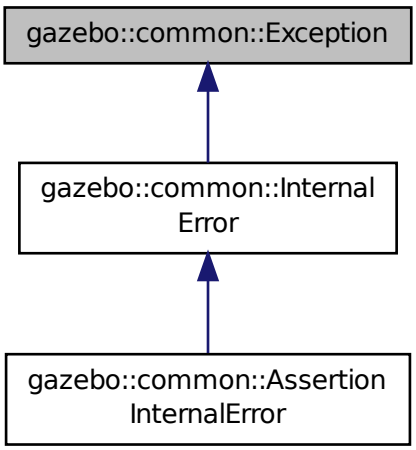
- **Event.hh**

## 10.66 gazebo::common::Exception Class Reference

Class for generating exceptions.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Exception:



## Public Member Functions

- **Exception** ()

  *Constructor.*
- **Exception** (const char *_file, int _line, std::string _msg)

  *Default constructor.*
- virtual ~**Exception** ()

  *Destructor.*
- std::string **GetErrorFile** () const

  *Return the error function.*
- std::string **GetErrorStr** () const

  *Return the error string.*
- void **Print** () const

  *Print the exception to std out.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Exception** &_err)

  *stream insertion operator for Gazebo Error*

### 10.66.1 Detailed Description

Class for generating exceptions.

### 10.66.2 Constructor & Destructor Documentation

#### 10.66.2.1 gazebo::common::Exception::Exception ( )

Constructor.

#### 10.66.2.2 gazebo::common::Exception::Exception ( const char * *_file*, int *_line*, std::string *_msg* )

Default constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_file* | File name |
| in | *_line* | Line number where the error occurred |
| in | *_msg* | Error message |

#### 10.66.2.3 virtual gazebo::common::Exception::~Exception ( ) [virtual]

Destructor.

### 10.66.3 Member Function Documentation

#### 10.66.3.1 std::string gazebo::common::Exception::GetErrorFile ( ) const

Return the error function.

**Returns**

The error function name

#### 10.66.3.2 std::string gazebo::common::Exception::GetErrorStr ( ) const

Return the error string.

**Returns**

The error string

#### 10.66.3.3 void gazebo::common::Exception::Print ( ) const

Print the exception to std out.

### 10.66.4 Friends And Related Function Documentation

#### 10.66.4.1 std::ostream& operator<< ( std::ostream & *_out*, const gazebo::common::Exception & *_err* ) [friend]

stream insertion operator for Gazebo Error

**Parameters**

| | | |
|---|---|---|
| in | *_out* | the output stream |
| in | *_err* | the exception |

The documentation for this class was generated from the following file:

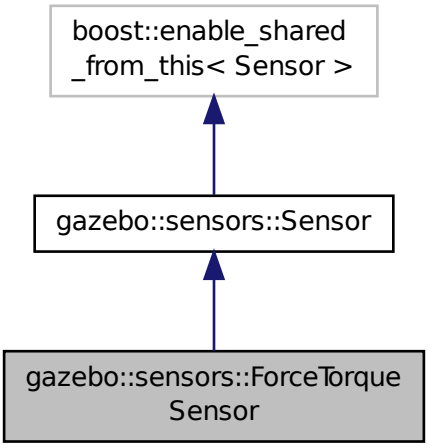
- **Exception.hh**

## 10.67 gazebo::sensors::ForceTorqueSensor Class Reference

**Sensor** (p. 837) for measure force and torque on a joint.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ForceTorqueSensor:




## Public Member Functions

- **ForceTorqueSensor** ()

  *Constructor.*
- virtual ~**ForceTorqueSensor** ()

  *Destructor.*
- template<typename T >
  **event::ConnectionPtr ConnectUpdate** (T _subscriber)

  *Connect a to the update signal.*
- void **DisconnectUpdate** (**event::ConnectionPtr** &_conn)

  *Disconnect from the update signal.*
- **math::Vector3 GetForce** () const

  *Get the current joint force.*
- **physics::JointPtr GetJoint** () const

  *Get Parent Joint.*
- virtual std::string **GetTopic** () const

  *Returns the topic name as set in SDF.*
- **math::Vector3 GetTorque** () const

  *Get the current joint torque.*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

### Protected Member Functions

- virtual void **Fini** ()

  *Finalize the sensor.*
- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Protected Attributes

- **event::EventT**< void(msgs::WrenchStamped)> **update**

  *Update event.*

## 10.67.1 Detailed Description

**Sensor** (p. 837) for measure force and torque on a joint.

## 10.67.2 Constructor & Destructor Documentation

### 10.67.2.1 gazebo::sensors::ForceTorqueSensor::ForceTorqueSensor ( )

Constructor.

### 10.67.2.2 virtual gazebo::sensors::ForceTorqueSensor::~ForceTorqueSensor ( ) [virtual]

Destructor.

## 10.67.3 Member Function Documentation

### 10.67.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::ForceTorqueSensor::ConnectUpdate ( T *_subscriber* ) [inline]

Connect a to the update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback function. |

**Returns**

The connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect(), and update.

### 10.67.3.2 void gazebo::sensors::ForceTorqueSensor::DisconnectUpdate ( event::ConnectionPtr & *_conn* ) [inline]

Disconnect from the update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | Connection to remove. |

References gazebo::event::EventT< T >::Disconnect(), and update.

#### 10.67.3.3 virtual void gazebo::sensors::ForceTorqueSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.67.3.4 math::Vector3 gazebo::sensors::ForceTorqueSensor::GetForce ( ) const

Get the current joint force.

**Returns**

The latested measured force.

#### 10.67.3.5 physics::JointPtr gazebo::sensors::ForceTorqueSensor::GetJoint ( ) const

Get Parent Joint.

**Returns**

Pointer to the joint containing this sensor

#### 10.67.3.6 virtual std::string gazebo::sensors::ForceTorqueSensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

**Returns**

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 843).

#### 10.67.3.7 math::Vector3 gazebo::sensors::ForceTorqueSensor::GetTorque ( ) const

Get the current joint torque.

**Returns**

The latested measured torque.

#### 10.67.3.8 virtual void gazebo::sensors::ForceTorqueSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.67.3.9 virtual bool gazebo::sensors::ForceTorqueSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.67.3.10 virtual void gazebo::sensors::ForceTorqueSensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.67.3.11 virtual void gazebo::sensors::ForceTorqueSensor::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

### 10.67.4 Member Data Documentation

#### 10.67.4.1 event::EventT<void(msgs::WrenchStamped)> gazebo::sensors::ForceTorqueSensor::update [protected]

Update event.

Referenced by ConnectUpdate(), and DisconnectUpdate().

The documentation for this class was generated from the following file:

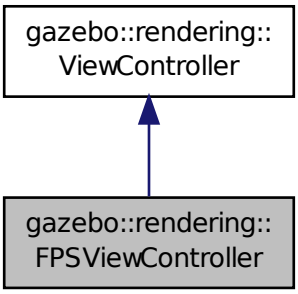
- **ForceTorqueSensor.hh**

## 10.68 gazebo::rendering::FPSViewController Class Reference

First Person Shooter style view controller.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::FPSViewController:




## Public Member Functions

- **FPSViewController** (**UserCameraPtr** _camera)

  *Constructor.*
- virtual ~**FPSViewController** ()

  *Destructor.*
- void **HandleKeyPressEvent** (const std::string &_key)

  *Handle a key press event.*
- void **HandleKeyReleaseEvent** (const std::string &_key)

  *Handle a key release event.*
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)

  *Handle a mouse event.*
- virtual void **Init** ()

  *Initialize the controller.*
- virtual void **Update** ()

  *Update the camera position.*

## Static Public Member Functions

- static std::string **GetTypeString** ()

  *Get the type name of this view controller.*

## Additional Inherited Members

### 10.68.1 Detailed Description

First Person Shooter style view controller.

### 10.68.2 Constructor & Destructor Documentation

#### 10.68.2.1 gazebo::rendering::FPSViewController::FPSViewController ( UserCameraPtr _camera )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | ***Camera*** *(p. 186)* | to controll |

#### 10.68.2.2 virtual gazebo::rendering::FPSViewController::~FPSViewController ( ) [virtual]

Destructor.

### 10.68.3 Member Function Documentation

#### 10.68.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

**Returns**

The name of the controller type: "fps"

#### 10.68.3.2 void gazebo::rendering::FPSViewController::HandleKeyPressEvent ( const std::string & _key ) [virtual]

Handle a key press event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key that was pressed. |

Implements **gazebo::rendering::ViewController** (p. 1119).

#### 10.68.3.3 void gazebo::rendering::FPSViewController::HandleKeyReleaseEvent ( const std::string & _key ) [virtual]

Handle a key release event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key that was released. |

Implements **gazebo::rendering::ViewController** (p. 1120).

#### 10.68.3.4 virtual void gazebo::rendering::FPSViewController::HandleMouseEvent ( const common::MouseEvent & _event ) [virtual]

Handle a mouse event.

**Parameters**

| | | |
|---|---|---|
| in | *_event* | The mouse position. |

Implements **gazebo::rendering::ViewController** (p. 1120).

#### 10.68.3.5 virtual void gazebo::rendering::FPSViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 1120).

#### 10.68.3.6 virtual void gazebo::rendering::FPSViewController::Update ( ) [virtual]

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 1121).

The documentation for this class was generated from the following file:

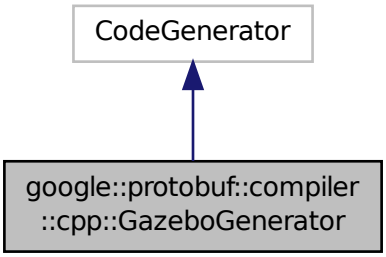
- **FPSViewController.hh**

## 10.69 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 108).

`#include <GazeboGenerator.hh>`

Inheritance diagram for google::protobuf::compiler::cpp::GazeboGenerator:




### Public Member Functions

- **GazeboGenerator** (const std::string &_name)
- virtual ~**GazeboGenerator** ()
- virtual bool **Generate** (const FileDescriptor *file, const string &parameter, OutputDirectory *directory, string *error) const

### 10.69.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 108).

### 10.69.2 Constructor & Destructor Documentation

#### 10.69.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator ( const std::string & *_name* )

#### 10.69.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~GazeboGenerator ( ) [virtual]

### 10.69.3 Member Function Documentation

#### 10.69.3.1 virtual bool google::protobuf::compiler::cpp::GazeboGenerator::Generate ( const FileDescriptor * *file*, const string & *parameter*, OutputDirectory * *directory*, string * *error* ) const [virtual]

The documentation for this class was generated from the following file:

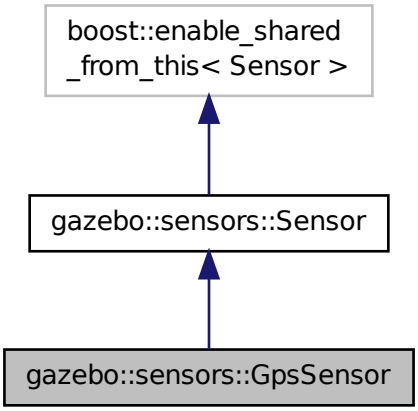
- **GazeboGenerator.hh**

## 10.70 gazebo::sensors::GpsSensor Class Reference

**GpsSensor** (p. 426) to provide position measurement.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::GpsSensor:




### Public Member Functions

- **GpsSensor** ()

*Constructor.*

- virtual ~**GpsSensor** ()

  *Destructor.*

- virtual void **Fini** ()

  *Finalize the sensor.*

- double **GetAltitude** () const

  *Accessor for current altitude.*

- **math::Angle GetLatitude** () const

  *Accessor for current latitude angle.*

- **math::Angle GetLongitude** () const

  *Accessor for current longitude angle.*

- virtual void **Init** ()

  *Initialize the sensor.*

- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr &_sdf)
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

### Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Additional Inherited Members

### 10.70.1 Detailed Description

**GpsSensor** (p. 426) to provide position measurement.

### 10.70.2 Constructor & Destructor Documentation

#### 10.70.2.1 gazebo::sensors::GpsSensor::GpsSensor ( )

Constructor.

#### 10.70.2.2 virtual gazebo::sensors::GpsSensor::~GpsSensor ( ) [virtual]

Destructor.

### 10.70.3 Member Function Documentation

#### 10.70.3.1 virtual void gazebo::sensors::GpsSensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.70.3.2 double gazebo::sensors::GpsSensor::GetAltitude ( ) const

Accessor for current altitude.

**Returns**

Current altitude above sea level.

#### 10.70.3.3 math::Angle gazebo::sensors::GpsSensor::GetLatitude ( ) const

Accessor for current latitude angle.

**Returns**

Current latitude angle.

#### 10.70.3.4 math::Angle gazebo::sensors::GpsSensor::GetLongitude ( ) const

Accessor for current longitude angle.

**Returns**

Current longitude angle.

#### 10.70.3.5 virtual void gazebo::sensors::GpsSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.70.3.6 virtual void gazebo::sensors::GpsSensor::Load ( const std::string & *_worldName*, sdf::ElementPtr & *_sdf* ) [virtual]

#### 10.70.3.7 virtual void gazebo::sensors::GpsSensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.70.3.8 virtual void gazebo::sensors::GpsSensor::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

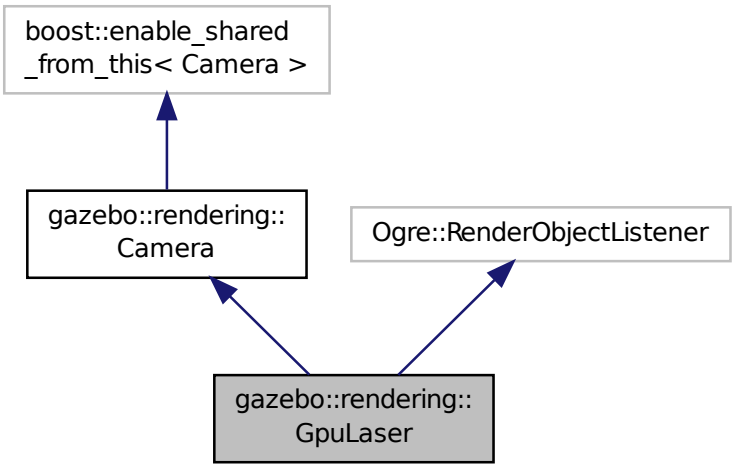
- **GpsSensor.hh**

## 10.71 gazebo::rendering::GpuLaser Class Reference

GPU based laser distance sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::GpuLaser:




### Public Member Functions

- **GpuLaser** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)

  *Constructor.*
- virtual ~**GpuLaser** ()

  *Destructor.*
- template<typename T >
  **event::ConnectionPtr ConnectNewLaserFrame** (T _subscriber)

  *Connect to a laser frame signal.*
- void **CreateLaserTexture** (const std::string &_textureName)

  *Create the texture which is used to render laser data.*
- void **DisconnectNewLaserFrame** (**event::ConnectionPtr** &_c)

*Disconnect from a laser frame signal.*

- virtual void **Fini** ()

  *Finalize the camera.*

- double **GetCameraCount** () const

  *Get the number of cameras required.*

- double **GetCosHorzFOV** () const

  *Get Cos Horz field-of-view.*

- double **GetCosVertFOV** () const

  *Get Cos Vert field-of-view.*

- double **GetFarClip** () const

  *Get far clip.*

- double **GetHorzFOV** () const

  *Get the horizontal field of view of the laser sensor.*

- double **GetHorzHalfAngle** () const

  *Get (horizontal_max_angle + horizontal_min_angle) ∗ 0.5.*

- const float ∗ **GetLaserData** ()

  *All things needed to get back z buffer for laser data.*

- double **GetNearClip** () const

  *Get near clip.*

- double **GetRayCountRatio** () const

  *Get the ray count ratio (equivalent to aspect ratio)*

- double **GetVertFOV** () const

  *Get the vertical field-of-view.*

- double **GetVertHalfAngle** () const

  *Get (vertical_max_angle + vertical_min_angle) ∗ 0.5.*

- virtual void **Init** ()

  *Initialize the camera.*

- bool **IsHorizontal** () const

  *Gets if sensor is horizontal.*

- virtual void **Load** (sdf::ElementPtr &_sdf)

- virtual void **Load** ()

  *Load the camera with default parmeters.*

- virtual void **notifyRenderSingleObject** (Ogre::Renderable ∗_rend, const Ogre::Pass ∗_p, const Ogre::AutoParamDataSource ∗_s, const Ogre::LightList ∗_ll, bool _supp)

- virtual void **PostRender** ()

  *Post render.*

- void **SetCameraCount** (double _cameraCount)

  *Set the number of cameras required.*

- void **SetCosHorzFOV** (double _chfov)

  *Set the Cos Horz FOV.*

- void **SetCosVertFOV** (double _cvfov)

  *Set the Cos Horz FOV.*

- void **SetFarClip** (double _far)

  *Set the far clip distance.*

- void **SetHorzFOV** (double _hfov)

  *Set the horizontal fov.*

- void **SetHorzHalfAngle** (double _angle)

*Set the horizontal half angle.*

- void **SetIsHorizontal** (bool _horizontal)

  *Set sensor horizontal or vertical.*

- void **SetNearClip** (double _near)

  *Set the near clip distance.*

- void **SetRangeCount** (unsigned int _w, unsigned int _h=1)

  *Set the number of laser samples in the width and height.*

- void **SetRayCountRatio** (double _rayCountRatio)

  *Sets the ray count ratio (equivalen to aspect ratio)*

- void **SetVertFOV** (double _vfov)

  *Set the vertical fov.*

- void **SetVertHalfAngle** (double _angle)

  *Set the vertical half angle.*

### Protected Attributes

- unsigned int **cameraCount**

  *Number of cameras needed to generate the rays.*

- double **chfov**

  *Cos horizontal field-of-view.*

- double **cvfov**

  *Cos vertical field-of-view.*

- double **far**

  *Far clip plane.*

- double **hfov**

  *Horizontal field-of-view.*

- double **horzHalfAngle**

  *Horizontal half angle.*

- bool **isHorizontal**

  *True if the sensor is horizontal only.*

- double **near**

  *Near clip plane.*

- double **rayCountRatio**

  *Ray count ratio.*

- double **vertHalfAngle**

  *Vertical half angle.*

- double **vfov**

  *Vertical field-of-view.*

### Additional Inherited Members

### 10.71.1 Detailed Description

GPU based laser distance sensor.

### 10.71.2 Constructor & Destructor Documentation

#### 10.71.2.1 gazebo::rendering::GpuLaser::GpuLaser ( const std::string & *_namePrefix,* **ScenePtr** *_scene,* bool *_autoRender =* `true` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_namePrefix* | Unique prefix name for the camera. |
| in | *_scene* | **Scene** (p. 814) that will contain the camera |
| in | *_autoRender* | Almost everyone should leave this as true. |

#### 10.71.2.2 virtual gazebo::rendering::GpuLaser::~GpuLaser ( ) `[virtual]`

Destructor.

### 10.71.3 Member Function Documentation

#### 10.71.3.1 template<typename T > event::ConnectionPtr gazebo::rendering::GpuLaser::ConnectNewLaserFrame ( T *_subscriber* ) `[inline]`

Connect to a laser frame signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback that is called when a new image is generated |

**Returns**

A pointer to the connection. This must be kept in scope.

References gazebo::event::EventT< T >::Connect().

#### 10.71.3.2 void gazebo::rendering::GpuLaser::CreateLaserTexture ( const std::string & *_textureName* )

Create the texture which is used to render laser data.

**Parameters**

| | | |
|---|---|---|
| in | *_textureName* | Name of the new texture. |

#### 10.71.3.3 void gazebo::rendering::GpuLaser::DisconnectNewLaserFrame ( event::ConnectionPtr & *_c* ) `[inline]`

Disconnect from a laser frame signal.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The connection to disconnect |

References gazebo::event::EventT< T >::Disconnect().

#### 10.71.3.4 virtual void gazebo::rendering::GpuLaser::Fini ( ) [virtual]

Finalize the camera.

This function is called before the camera is destructed

Reimplemented from **gazebo::rendering::Camera** (p. 195).

#### 10.71.3.5 double gazebo::rendering::GpuLaser::GetCameraCount ( ) const

Get the number of cameras required.

**Returns**

Number of cameras needed to generate the rays

#### 10.71.3.6 double gazebo::rendering::GpuLaser::GetCosHorzFOV ( ) const

Get Cos Horz field-of-view.

**Returns**

2 ∗ atan(tan(this->hfov/2) / cos(this->vfov/2))

#### 10.71.3.7 double gazebo::rendering::GpuLaser::GetCosVertFOV ( ) const

Get Cos Vert field-of-view.

**Returns**

2 ∗ atan(tan(this->vfov/2) / cos(this->hfov/2))

#### 10.71.3.8 double gazebo::rendering::GpuLaser::GetFarClip ( ) const

Get far clip.

**Returns**

far clip distance

#### 10.71.3.9 double gazebo::rendering::GpuLaser::GetHorzFOV ( ) const

Get the horizontal field of view of the laser sensor.

**Returns**

The horizontal field of view of the laser sensor.

#### 10.71.3.10 double gazebo::rendering::GpuLaser::GetHorzHalfAngle ( ) const

Get (horizontal_max_angle + horizontal_min_angle) ∗ 0.5.

**Returns**

(horizontal_max_angle + horizontal_min_angle) ∗ 0.5

#### 10.71.3.11 const float∗ gazebo::rendering::GpuLaser::GetLaserData ( )

All things needed to get back z buffer for laser data.

**Returns**

Array of laser data.

#### 10.71.3.12 double gazebo::rendering::GpuLaser::GetNearClip ( ) const

Get near clip.

**Returns**

near clip distance

#### 10.71.3.13 double gazebo::rendering::GpuLaser::GetRayCountRatio ( ) const

Get the ray count ratio (equivalent to aspect ratio)

**Returns**

The ray count ratio (equivalent to aspect ratio)

#### 10.71.3.14 double gazebo::rendering::GpuLaser::GetVertFOV ( ) const

Get the vertical field-of-view.

**Returns**

The vertical field of view of the laser sensor.

#### 10.71.3.15 double gazebo::rendering::GpuLaser::GetVertHalfAngle ( ) const

Get (vertical_max_angle + vertical_min_angle) ∗ 0.5.

**Returns**

(vertical_max_angle + vertical_min_angle) ∗ 0.5

#### 10.71.3.16 virtual void gazebo::rendering::GpuLaser::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 203).

#### 10.71.3.17 bool gazebo::rendering::GpuLaser::IsHorizontal ( ) const

Gets if sensor is horizontal.

**Returns**

True if horizontal, false if not

#### 10.71.3.18 virtual void gazebo::rendering::GpuLaser::Load ( sdf::ElementPtr & *_sdf* ) [virtual]

#### 10.71.3.19 virtual void gazebo::rendering::GpuLaser::Load ( ) [virtual]

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.71.3.20 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject ( Ogre::Renderable * *_rend*, const Ogre::Pass * *_p*, const Ogre::AutoParamDataSource * *_s*, const Ogre::LightList * *_ll*, bool *_supp* ) [virtual]

#### 10.71.3.21 virtual void gazebo::rendering::GpuLaser::PostRender ( ) [virtual]

Post render.

Called afer the render signal.

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.71.3.22 void gazebo::rendering::GpuLaser::SetCameraCount ( double *_cameraCount* )

Set the number of cameras required.

**Parameters**

| | | |
|---|---|---|
| in | *_cameraCount* | The number of cameras required to generate the rays |

#### 10.71.3.23 void gazebo::rendering::GpuLaser::SetCosHorzFOV ( double *_chfov* )

Set the Cos Horz FOV.

**Parameters**

| | | |
|---|---|---|
| in | *_chfov* | Cos Horz FOV |

#### 10.71.3.24 void gazebo::rendering::GpuLaser::SetCosVertFOV ( double *_cvfov* )

Set the Cos Horz FOV.

**Parameters**

| | | |
|---|---|---|
| in | *_cvfov* | Cos Horz FOV |

#### 10.71.3.25 void gazebo::rendering::GpuLaser::SetFarClip ( double *_far* )

Set the far clip distance.

**Parameters**

| | | |
|---|---|---|
| in | *_far* | far clip distance |

#### 10.71.3.26 void gazebo::rendering::GpuLaser::SetHorzFOV ( double *_hfov* )

Set the horizontal fov.

**Parameters**

| | | |
|---|---|---|
| in | *_hfov* | horizontal fov |

#### 10.71.3.27 void gazebo::rendering::GpuLaser::SetHorzHalfAngle ( double *_angle* )

Set the horizontal half angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | horizontal half angle |

#### 10.71.3.28 void gazebo::rendering::GpuLaser::SetIsHorizontal ( bool *_horizontal* )

Set sensor horizontal or vertical.

**Parameters**

| | | |
|---|---|---|
| in | *_horizontal* | True if horizontal, false if not |

#### 10.71.3.29 void gazebo::rendering::GpuLaser::SetNearClip ( double *_near* )

Set the near clip distance.

**Parameters**

| | | |
|---|---|---|
| in | *_near* | near clip distance |

#### 10.71.3.30 void gazebo::rendering::GpuLaser::SetRangeCount ( unsigned int _w, unsigned int _h = 1 )

Set the number of laser samples in the width and height.

**Parameters**

| | | |
|---|---|---|
| in | *_w* | Number of samples in the horizontal sweep |
| in | *_h* | Number of samples in the vertical sweep |

#### 10.71.3.31 void gazebo::rendering::GpuLaser::SetRayCountRatio ( double _rayCountRatio )

Sets the ray count ratio (equivalen to aspect ratio)

**Parameters**

| | | |
|---|---|---|
| in | *_rayCountRatio* | ray count ratio (equivalent to aspect ratio) |

#### 10.71.3.32 void gazebo::rendering::GpuLaser::SetVertFOV ( double _vfov )

Set the vertical fov.

**Parameters**

| | | |
|---|---|---|
| in | *_vfov* | vertical fov |

#### 10.71.3.33 void gazebo::rendering::GpuLaser::SetVertHalfAngle ( double _angle )

Set the vertical half angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | vertical half angle |

### 10.71.4 Member Data Documentation

#### 10.71.4.1 unsigned int gazebo::rendering::GpuLaser::cameraCount [protected]

Number of cameras needed to generate the rays.

#### 10.71.4.2 double gazebo::rendering::GpuLaser::chfov [protected]

Cos horizontal field-of-view.

#### 10.71.4.3 double gazebo::rendering::GpuLaser::cvfov [protected]

Cos vertical field-of-view.

#### 10.71.4.4 double gazebo::rendering::GpuLaser::far [protected]

Far clip plane.

#### 10.71.4.5 double gazebo::rendering::GpuLaser::hfov [protected]

Horizontal field-of-view.

#### 10.71.4.6 double gazebo::rendering::GpuLaser::horzHalfAngle [protected]

Horizontal half angle.

#### 10.71.4.7 bool gazebo::rendering::GpuLaser::isHorizontal [protected]

True if the sensor is horizontal only.

#### 10.71.4.8 double gazebo::rendering::GpuLaser::near [protected]

Near clip plane.

#### 10.71.4.9 double gazebo::rendering::GpuLaser::rayCountRatio [protected]

Ray count ratio.

#### 10.71.4.10 double gazebo::rendering::GpuLaser::vertHalfAngle [protected]

Vertical half angle.

#### 10.71.4.11 double gazebo::rendering::GpuLaser::vfov [protected]

Vertical field-of-view.

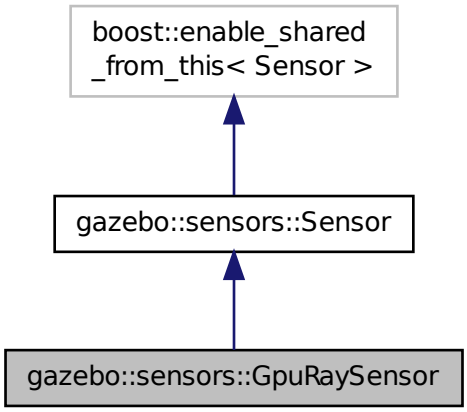
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

## 10.72 gazebo::sensors::GpuRaySensor Class Reference

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::GpuRaySensor:




## Public Member Functions

- **GpuRaySensor** ()

  *Constructor.*
- virtual ∼**GpuRaySensor** ()

  *Destructor.*
- **event::ConnectionPtr ConnectNewLaserFrame** (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> _subscriber)

  *Connect to the new laser frame event.*
- void **DisconnectNewLaserFrame** (**event::ConnectionPtr** &_conn)

  *Disconnect Laser Frame.*
- **math::Angle GetAngleMax** () const

  *Get the maximum angle.*
- **math::Angle GetAngleMin** () const

  *Get the minimum angle.*
- double **GetAngleResolution** () const

  *Get radians between each range.*
- unsigned int **GetCameraCount** () const

  *Gets the camera count.*
- double **GetCosHorzFOV** () const

  *Get Cos Horz field-of-view.*
- double **GetCosVertFOV** () const

  *Get Cos Vert field-of-view.*
- int **GetFiducial** (int _index) const

  *Get detected fiducial value for a ray.*
- double **GetHorzFOV** () const

*Get the horizontal field of view of the laser sensor.*

- double **GetHorzHalfAngle** () const

  *Get (horizontal_max_angle + horizontal_min_angle) ∗ 0.5.*

- **rendering::GpuLaserPtr GetLaserCamera** () const

  *Returns a pointer to the internally kept* ***rendering::GpuLaser*** *(*p. *429**).*

- double **GetRange** (int _index)

  *Get detected range for a ray.*

- int **GetRangeCount** () const

  *Get the range count.*

- double **GetRangeCountRatio** () const

  *Return the ratio of horizontal range count to vertical range count.*

- double **GetRangeMax** () const

  *Get the maximum range.*

- double **GetRangeMin** () const

  *Get the minimum range.*

- double **GetRangeResolution** () const

  *Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.*

- void **GetRanges** (std::vector< double > &_ranges)

  *Get all the ranges.*

- int **GetRayCount** () const

  *Get the ray count.*

- double **GetRayCountRatio** () const

  *Return the ratio of horizontal ray count to vertical ray count.*

- double **GetRetro** (int _index) const

  *Get detected retro (intensity) value for a ray.*

- virtual std::string **GetTopic** () const

  *Returns the topic name as set in SDF.*

- double **GetVertFOV** () const

  *Get the vertical field-of-view.*

- double **GetVertHalfAngle** () const

  *Get (vertical_max_angle + vertical_min_angle) ∗ 0.5.*

- **math::Angle GetVerticalAngleMax** () const

  *Get the vertical scan line top angle.*

- **math::Angle GetVerticalAngleMin** () const

  *Get the vertical scan bottom angle.*

- int **GetVerticalRangeCount** () const

  *Get the vertical scan line count.*

- int **GetVerticalRayCount** () const

  *Get the vertical scan line count.*

- virtual void **Init** ()

  *Initialize the ray.*

- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*

- bool **IsHorizontal** () const

  *Gets if sensor is horizontal.*

- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr &_sdf)

*Load the sensor with SDF parameters.*

- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*
- void **SetAngleMax** (double _angle)

  *Set the scan maximum angle.*
- void **SetAngleMin** (double _angle)

  *Set the scan minimum angle.*
- void **SetVerticalAngleMax** (double _angle)

  *Set the vertical scan line top angle.*
- void **SetVerticalAngleMin** (double _angle)

  *Set the vertical scan bottom angle.*

### Protected Member Functions

- virtual void **Fini** ()

  *Finalize the ray.*
- virtual void **UpdateImpl** (bool _force)

  *Update the sensor information.*

### Protected Attributes

- sdf::ElementPtr **cameraElem**

  *Camera SDF element.*
- sdf::ElementPtr **horzElem**

  *Horizontal SDF element.*
- unsigned int **horzRangeCount**

  *Horizontal range count.*
- unsigned int **horzRayCount**

  *Horizontal ray count.*
- double **rangeCountRatio**

  *Range count ratio.*
- sdf::ElementPtr **rangeElem**

  *Range SDF element.*
- sdf::ElementPtr **scanElem**

  *Scan SDF elementz.*
- sdf::ElementPtr **vertElem**

  *Vertical SDF element.*
- unsigned int **vertRangeCount**

  *Vertical range count.*
- unsigned int **vertRayCount**

  *Vertical ray count.*

## 10.72.1 Constructor & Destructor Documentation

#### 10.72.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ( )

Constructor.

#### 10.72.1.2 virtual gazebo::sensors::GpuRaySensor::~GpuRaySensor ( ) [virtual]

Destructor.

### 10.72.2 Member Function Documentation

#### 10.72.2.1 event::ConnectionPtr gazebo::sensors::GpuRaySensor::ConnectNewLaserFrame ( boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> _subscriber )

Connect to the new laser frame event.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Event callback. |

#### 10.72.2.2 void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame ( event::ConnectionPtr & _conn )

Disconnect Laser Frame.

**Parameters**

| | | |
|---|---|---|
| in,out | *_conn* | Connection pointer to disconnect. |

#### 10.72.2.3 virtual void gazebo::sensors::GpuRaySensor::Fini ( ) [protected], [virtual]

Finalize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.72.2.4 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax ( ) const

Get the maximum angle.

**Returns**

the maximum angle

#### 10.72.2.5 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin ( ) const

Get the minimum angle.

**Returns**

The minimum angle

#### 10.72.2.6 double gazebo::sensors::GpuRaySensor::GetAngleResolution ( ) const

Get radians between each range.

#### 10.72.2.7 unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount ( ) const

Gets the camera count.

**Returns**

Number of cameras

#### 10.72.2.8 double gazebo::sensors::GpuRaySensor::GetCosHorzFOV ( ) const

Get Cos Horz field-of-view.

**Returns**

2 ∗ atan(tan(this->hfov/2) / cos(this->vfov/2))

#### 10.72.2.9 double gazebo::sensors::GpuRaySensor::GetCosVertFOV ( ) const

Get Cos Vert field-of-view.

**Returns**

2 ∗ atan(tan(this->vfov/2) / cos(this->hfov/2))

#### 10.72.2.10 int gazebo::sensors::GpuRaySensor::GetFiducial ( int *_index* ) const

Get detected fiducial value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of specific ray |

**Returns**

Fiducial value of ray

#### 10.72.2.11 double gazebo::sensors::GpuRaySensor::GetHorzFOV ( ) const

Get the horizontal field of view of the laser sensor.

**Returns**

The horizontal field of view of the laser sensor.

#### 10.72.2.12 double gazebo::sensors::GpuRaySensor::GetHorzHalfAngle ( ) const

Get (horizontal_max_angle + horizontal_min_angle) ∗ 0.5.

**Returns**

(horizontal_max_angle + horizontal_min_angle) ∗ 0.5

#### 10.72.2.13 rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera ( ) const [inline]

Returns a pointer to the internally kept **rendering::GpuLaser** (p. 429).

**Returns**

Pointer to GpuLaser

#### 10.72.2.14 double gazebo::sensors::GpuRaySensor::GetRange ( int *_index* )

Get detected range for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of specific ray |

**Returns**

Returns DBL_MAX for no detection.

#### 10.72.2.15 int gazebo::sensors::GpuRaySensor::GetRangeCount ( ) const

Get the range count.

**Returns**

The number of ranges

#### 10.72.2.16 double gazebo::sensors::GpuRaySensor::GetRangeCountRatio ( ) const

Return the ratio of horizontal range count to vertical range count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

#### 10.72.2.17 double gazebo::sensors::GpuRaySensor::GetRangeMax ( ) const

Get the maximum range.

**Returns**

The maximum range

#### 10.72.2.18 double gazebo::sensors::GpuRaySensor::GetRangeMin ( ) const

Get the minimum range.

**Returns**

The minimum range

#### 10.72.2.19 double gazebo::sensors::GpuRaySensor::GetRangeResolution ( ) const

Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.

If it's less than 1, fewer simulated rays than actual returned range readings are used, the results are interpolated from two nearest neighbors, and vice versa.

**Returns**

The Range Resolution

#### 10.72.2.20 void gazebo::sensors::GpuRaySensor::GetRanges ( std::vector< double > & *_ranges* )

Get all the ranges.

**Parameters**

| | | |
|---|---|---|
| out | *_range* | A vector that will contain all the range data |

#### 10.72.2.21 int gazebo::sensors::GpuRaySensor::GetRayCount ( ) const

Get the ray count.

**Returns**

The number of rays

#### 10.72.2.22 double gazebo::sensors::GpuRaySensor::GetRayCountRatio ( ) const

Return the ratio of horizontal ray count to vertical ray count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

#### 10.72.2.23 double gazebo::sensors::GpuRaySensor::GetRetro ( int *_index* ) const

Get detected retro (intensity) value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of specific ray |

**Returns**

Intensity value of ray

#### 10.72.2.24 virtual std::string gazebo::sensors::GpuRaySensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

**Returns**

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 843).

#### 10.72.2.25 double gazebo::sensors::GpuRaySensor::GetVertFOV ( ) const

Get the vertical field-of-view.

#### 10.72.2.26 double gazebo::sensors::GpuRaySensor::GetVertHalfAngle ( ) const

Get (vertical_max_angle + vertical_min_angle) $*$ 0.5.

**Returns**

(vertical_max_angle + vertical_min_angle) $*$ 0.5

#### 10.72.2.27 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

**Returns**

The Maximum angle of the scan block

#### 10.72.2.28 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

**Returns**

The minimum angle of the scan block

#### 10.72.2.29 int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

**Returns**

The number of scan lines vertically

#### 10.72.2.30 int gazebo::sensors::GpuRaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

**Returns**

The number of scan lines vertically

#### 10.72.2.31 virtual void gazebo::sensors::GpuRaySensor::Init ( ) [virtual]

Initialize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.72.2.32 virtual bool gazebo::sensors::GpuRaySensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.72.2.33 bool gazebo::sensors::GpuRaySensor::IsHorizontal ( ) const

Gets if sensor is horizontal.

**Returns**

True if horizontal, false if not

#### 10.72.2.34 virtual void gazebo::sensors::GpuRaySensor::Load ( const std::string & _worldName, sdf::ElementPtr & _sdf ) [virtual]

Load the sensor with SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF **Sensor** (p. 837) parameters |
| in | *_worldName* | Name of world to load from |

#### 10.72.2.35 virtual void gazebo::sensors::GpuRaySensor::Load ( const std::string & _worldName ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.72.2.36 void gazebo::sensors::GpuRaySensor::SetAngleMax ( double _angle )

Set the scan maximum angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | The maximum angle |

#### 10.72.2.37 void gazebo::sensors::GpuRaySensor::SetAngleMin ( double _angle )

Set the scan minimum angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | The minimum angle |

#### 10.72.2.38 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax ( double _angle )

Set the vertical scan line top angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | The Maximum angle of the scan block |

#### 10.72.2.39 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin ( double _angle )

Set the vertical scan bottom angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | The minimum angle of the scan block |

#### 10.72.2.40 virtual void gazebo::sensors::GpuRaySensor::UpdateImpl ( bool *_force* ) [protected], [virtual]

Update the sensor information.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

### 10.72.3 Member Data Documentation

#### 10.72.3.1 sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem [protected]

Camera SDF element.

#### 10.72.3.2 sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem [protected]

Horizontal SDF element.

#### 10.72.3.3 unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount [protected]

Horizontal range count.

#### 10.72.3.4 unsigned int gazebo::sensors::GpuRaySensor::horzRayCount [protected]

Horizontal ray count.

#### 10.72.3.5 double gazebo::sensors::GpuRaySensor::rangeCountRatio [protected]

Range count ratio.

#### 10.72.3.6 sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem [protected]

Range SDF element.

#### 10.72.3.7 sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem [protected]

Scan SDF elementz.

#### 10.72.3.8 sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem [protected]

Vertical SDF element.

#### 10.72.3.9 unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount [protected]

Vertical range count.

#### 10.72.3.10 unsigned int gazebo::sensors::GpuRaySensor::vertRayCount [protected]

Vertical ray count.

The documentation for this class was generated from the following file:

- **GpuRaySensor.hh**

## 10.73 gazebo::rendering::Grid Class Reference

Displays a grid of cells, drawn with lines.

`#include <rendering/rendering.hh>`

### Public Member Functions

- **Grid** (**Scene** *_scene, uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)
  *Constructor.*
- ~**Grid** ()
  *Destructor.*
- void **Enable** (bool _enable)
  *Enable or disable the grid.*
- uint32_t **GetCellCount** () const
  *Get the number of cells.*
- float **GetCellLength** () const
  *Get the cell length.*
- **common::Color GetColor** () const
  *Return the grid color.*
- uint32_t **GetHeight** () const
  *Get the height of the grid.*
- float **GetLineWidth** () const
  *Get the width of the grid line.*
- Ogre::SceneNode * **GetSceneNode** ()
  *Get the **Ogre** (p. 130) scene node associated with this grid.*
- void **Init** ()
  *Initialize the grid.*
- void **SetCellCount** (uint32_t _count)
  *Set the number of cells.*
- void **SetCellLength** (float _len)
  *Set the cell length.*
- void **SetColor** (const **common::Color** &_color)
  *Sets the color of the grid.*
- void **SetHeight** (uint32_t _count)

*Set the height of the grid.*

- void **SetLineWidth** (float _width)

  *Set the line width.*

- void **SetUserData** (const Ogre::Any &_data)

  *Sets user data on all ogre objects we own.*

### 10.73.1 Detailed Description

Displays a grid of cells, drawn with lines.

Displays a grid of cells, drawn with lines. A grid with an identity orientation is drawn along the XY plane.

### 10.73.2 Constructor & Destructor Documentation

#### 10.73.2.1 gazebo::rendering::Grid::Grid ( Scene * *_scene,* uint32_t *_cellCount,* float *_cellLength,* float *_lineWidth,* const common::Color & *_color* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | The scene this object is part of |
| in | *_cellCount* | The number of cells to draw |
| in | *_cellLength* | The size of each cell |
| in | *_lineWidth* | The width of the lines to use |
| in | *_color* | The color of the grid |

#### 10.73.2.2 gazebo::rendering::Grid::~Grid ( )

Destructor.

### 10.73.3 Member Function Documentation

#### 10.73.3.1 void gazebo::rendering::Grid::Enable ( bool *_enable* )

Enable or disable the grid.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | Set to true to view the grid, false to make invisible. |

#### 10.73.3.2 uint32_t gazebo::rendering::Grid::GetCellCount ( ) const [inline]

Get the number of cells.

#### 10.73.3.3 float gazebo::rendering::Grid::GetCellLength ( ) const [inline]

Get the cell length.

**Returns**

The cell length

#### 10.73.3.4 common::Color gazebo::rendering::Grid::GetColor ( ) const [inline]

Return the grid color.

**Returns**

The grid color

#### 10.73.3.5 uint32_t gazebo::rendering::Grid::GetHeight ( ) const [inline]

Get the height of the grid.

**Returns**

The height

#### 10.73.3.6 float gazebo::rendering::Grid::GetLineWidth ( ) const [inline]

Get the width of the grid line.

**Returns**

The line width

#### 10.73.3.7 Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode ( ) [inline]

Get the **Ogre** (p. 130) scene node associated with this grid.

**Returns**

The **Ogre** (p. 130) scene node associated with this grid

#### 10.73.3.8 void gazebo::rendering::Grid::Init ( )

Initialize the grid.

#### 10.73.3.9 void gazebo::rendering::Grid::SetCellCount ( uint32_t *_count* )

Set the number of cells.

**Parameters**

| | | |
|---|---|---|
| in | *_count* | The number of cells |

#### 10.73.3.10 void gazebo::rendering::Grid::SetCellLength ( float *_len* )

Set the cell length.

**Parameters**

| | | |
|---|---|---|
| in | *_len* | The cell length |

#### 10.73.3.11 void gazebo::rendering::Grid::SetColor ( const common::Color & *_color* )

Sets the color of the grid.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | The grid color |

#### 10.73.3.12 void gazebo::rendering::Grid::SetHeight ( uint32_t *_count* )

Set the height of the grid.

**Parameters**

| | | |
|---|---|---|
| in | *_count* | **Grid** (p. 450) height |

#### 10.73.3.13 void gazebo::rendering::Grid::SetLineWidth ( float *_width* )

Set the line width.

**Parameters**

| | | |
|---|---|---|
| in | *_width* | The width of the grid |

#### 10.73.3.14 void gazebo::rendering::Grid::SetUserData ( const Ogre::Any & *_data* )

Sets user data on all ogre objects we own.

**Parameters**

| | | |
|---|---|---|
| in | *_data* | The user data |

The documentation for this class was generated from the following file:

- **Grid.hh**

## 10.74 gazebo::physics::Gripper Class Reference

A gripper abstraction.

`#include <physics/physics.hh>`

### Public Member Functions

- **Gripper** (**ModelPtr** _model)

  *Constructor.*
- virtual ~**Gripper** ()

  *Destructor.*
- std::string **GetName** () const

  *Return the name of the gripper.*
- virtual void **Init** ()

  *Initialize.*
- bool **IsAttached** () const

  *True if the gripper is attached to another model.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the gripper.*

### Protected Attributes

- **transport::NodePtr node**

  *Node for communication.*

## 10.74.1 Detailed Description

A gripper abstraction.

A gripper is a collection of links that act as a gripper. This class will intelligently generate fixed joints between the gripper and an object within the gripper. This allows the object to be manipulated without falling or behaving poorly.

## 10.74.2 Constructor & Destructor Documentation

### 10.74.2.1 gazebo::physics::Gripper::Gripper ( **ModelPtr** *_model* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | The model which contains the **Gripper** (p. 453). |

### 10.74.2.2 virtual gazebo::physics::Gripper::~Gripper ( ) [virtual]

Destructor.

## 10.74.3 Member Function Documentation

### 10.74.3.1 std::string gazebo::physics::Gripper::GetName ( ) const

Return the name of the gripper.

#### 10.74.3.2 virtual void gazebo::physics::Gripper::Init ( ) [virtual]

Initialize.

#### 10.74.3.3 bool gazebo::physics::Gripper::IsAttached ( ) const

True if the gripper is attached to another model.

**Returns**

True if the gripper is active and a joint has been created between the gripper and another model.

#### 10.74.3.4 virtual void gazebo::physics::Gripper::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the gripper.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Shared point to an sdf element that contains the list of links in the gripper. |

### 10.74.4 Member Data Documentation

#### 10.74.4.1 transport::NodePtr gazebo::physics::Gripper::node [protected]

Node for communication.

The documentation for this class was generated from the following file:

- **Gripper.hh**

## 10.75 gazebo::rendering::GUIOverlay Class Reference

A class that creates a CEGUI overlay on a render window.

`#include <rendering/rendering.hh>`

### Public Member Functions

- **GUIOverlay** ()

  *Constructor.*
- virtual ~**GUIOverlay** ()

  *Destructor.*
- bool **AttachCameraToImage** (**CameraPtr** &_camera, const std::string &_windowName)

  *Use this function to draw the output from a **rendering::Camera*** (p. *186*) *to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)

  *Use this function to draw the output from a **rendering::DepthCamera*** (p. *357*) *to and overlay window.*
- template<typename T >
  void **ButtonCallback** (const std::string &_buttonName, void(T::*_fp)(), T *_obj)

*Register a CEGUI button callback.*

- void **CreateWindow** (const std::string &_type, const std::string &_name, const std::string &_parent, const **math::Vector2d** &_position, const **math::Vector2d** &_size, const std::string &_text)

  *Create a new window on the overlay.*
- bool **HandleKeyPressEvent** (const std::string &_key)

  *Handle a key press event.*
- bool **HandleKeyReleaseEvent** (const std::string &_key)

  *Handle a key release event.*
- bool **HandleMouseEvent** (const **common::MouseEvent** &_evt)

  *Handle a mouse event.*
- void **Hide** ()

  *Make the overlay invisible.*
- void **Init** (Ogre::RenderTarget *_renderTarget)

  *Initialize the overlay.*
- bool **IsInitialized** ()

  *Return true if the overlay has been initialized.*
- void **LoadLayout** (const std::string &_filename)

  *Load a CEGUI layout file.*
- void **Resize** (unsigned int _width, unsigned int _height)

  *Resize the window.*
- void **Show** ()

  *Make the overlay visible.*
- void **Update** ()

  *Update the overlay's objects.*

### 10.75.1 Detailed Description

A class that creates a CEGUI overlay on a render window.

### 10.75.2 Constructor & Destructor Documentation

#### 10.75.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ( )

Constructor.

#### 10.75.2.2 virtual gazebo::rendering::GUIOverlay::~GUIOverlay ( ) [virtual]

Destructor.

### 10.75.3 Member Function Documentation

#### 10.75.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage ( CameraPtr & *_camera*, const std::string & *_windowName* )

Use this function to draw the output from a **rendering::Camera** (p. 186) to and overlay window.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | Pointer to the camera. |
| in | *_windowName* | Name of the window to receive the camera image |

**Returns**

True if successful

#### 10.75.3.2 bool gazebo::rendering::GUIOverlay::AttachCameraToImage ( DepthCameraPtr & *_camera*, const std::string & *_windowName* )

Use this function to draw the output from a **rendering::DepthCamera** (p. 357) to and overlay window.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | Pointer to the camera. |
| in | *_windowName* | Name of the window to receive the camera image |

**Returns**

True if successful

#### 10.75.3.3 template<typename T > void gazebo::rendering::GUIOverlay::ButtonCallback ( const std::string & *_buttonName*, void(T::*)() *_fp*, T * *_obj* ) [inline]

Register a CEGUI button callback.

Assign a callback to a name button.

**Parameters**

| | | |
|---|---|---|
| in | *_buttonName* | Name of the button. |
| in | *_fp* | Function pointer to the callback. |
| in | *_obj* | Class pointer that contains _fp. |

#### 10.75.3.4 void gazebo::rendering::GUIOverlay::CreateWindow ( const std::string & *_type*, const std::string & *_name*, const std::string & *_parent*, const math::Vector2d & *_position*, const math::Vector2d & *_size*, const std::string & *_text* )

Create a new window on the overlay.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | The window type. This should match a CEGUI window type. See CEGUI::WindowManager::getSingleton().createWindow(). |
| in | *_name* | Unique name for the window. |
| in | *_parent* | Name of the parent window. |
| in | *_position* | Position of the window within the parent. |
| in | *_size* | Size of the window. |
| in | *_text* | Display title of the window. |

#### 10.75.3.5 bool gazebo::rendering::GUIOverlay::HandleKeyPressEvent ( const std::string & _key )

Handle a key press event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key pressed. |

**Returns**

True if the key press event was handled.

#### 10.75.3.6 bool gazebo::rendering::GUIOverlay::HandleKeyReleaseEvent ( const std::string & _key )

Handle a key release event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key released. |

**Returns**

True if the key release event was handled.

#### 10.75.3.7 bool gazebo::rendering::GUIOverlay::HandleMouseEvent ( const common::MouseEvent & _evt )

Handle a mouse event.

**Parameters**

| | | |
|---|---|---|
| in | *_evt* | The mouse event. |

**Returns**

True if the mouse event was handled.

#### 10.75.3.8 void gazebo::rendering::GUIOverlay::Hide ( )

Make the overlay invisible.

#### 10.75.3.9 void gazebo::rendering::GUIOverlay::Init ( Ogre::RenderTarget * _renderTarget )

Initialize the overlay.

**Parameters**

| | | |
|---|---|---|
| in | *_renderTarget* | The render target which will have the overlay. |

#### 10.75.3.10 bool gazebo::rendering::GUIOverlay::IsInitialized ( )

Return true if the overlay has been initialized.

**Returns**

True if initialized

#### 10.75.3.11 void gazebo::rendering::GUIOverlay::LoadLayout ( const std::string & _filename )

Load a CEGUI layout file.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | Name of the layout file. |

#### 10.75.3.12 void gazebo::rendering::GUIOverlay::Resize ( unsigned int _width, unsigned int _height )

Resize the window.

#### 10.75.3.13 void gazebo::rendering::GUIOverlay::Show ( )

Make the overlay visible.

#### 10.75.3.14 void gazebo::rendering::GUIOverlay::Update ( )

Update the overlay's objects.

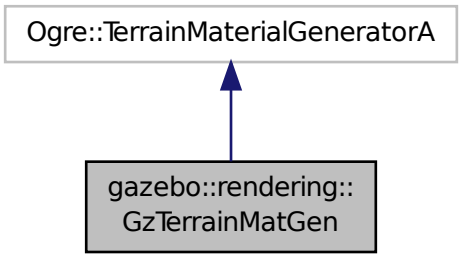
The documentation for this class was generated from the following file:

- **GUIOverlay.hh**

## 10.76 gazebo::rendering::GzTerrainMatGen Class Reference

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen:




### Classes

- class **SM2Profile**
  *Shader model 2 profile target.*

### Public Member Functions

- **GzTerrainMatGen** ()
  *Constructor.*
- virtual ~**GzTerrainMatGen** ()
  *Destructor.*

### 10.76.1 Constructor & Destructor Documentation

#### 10.76.1.1 gazebo::rendering::GzTerrainMatGen::GzTerrainMatGen ( )

Constructor.

#### 10.76.1.2 virtual gazebo::rendering::GzTerrainMatGen::~GzTerrainMatGen ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Heightmap.hh**

## 10.77 gazebo::rendering::Heightmap Class Reference

Rendering a terrain using heightmap information.

`#include <rendering/rendering.hh>`

## Public Member Functions

- **Heightmap** (**ScenePtr** _scene)

  *Constructor.*
- virtual ∼**Heightmap** ()

  *Destructor.*
- bool **Flatten** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  *Flatten the terrain based on a mouse press.*
- double **GetAvgHeight** (Ogre::Vector3 _pos, double _brushSize)

  *Get the average height around a point.*
- double **GetHeight** (double _x, double _y, double _z=1000)

  *Get the height at a location.*
- **common::Image GetImage** () const

  *Get the heightmap as an image.*
- Ogre::TerrainGroup::RayResult **GetMouseHit** (**CameraPtr** _camera, **math::Vector2i** _mousePos)

  *Calculate a mouse ray hit on the terrain.*
- Ogre::TerrainGroup ∗ **GetOgreTerrain** () const

  *Get a pointer to the OGRE terrain group object.*
- unsigned int **GetTerrainSubdivisionCount** () const

  *Get the number of subdivision the terrain will be split into.*
- void **Load** ()

  *Load the heightmap.*
- void **LoadFromMsg** (ConstVisualPtr &_msg)

  *Load the heightmap from a visual message.*
- bool **Lower** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  *Lower the terrain based on a mouse press.*
- bool **Raise** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  *Raise the terrain based on a mouse press.*
- void **SetWireframe** (bool _show)

  *Set the heightmap to render in wireframe mode.*
- bool **Smooth** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  *Smooth the terrain based on a mouse press.*
- void **SplitHeights** (const std::vector< float > &_heightmap, int _n, std::vector< std::vector< float > > &_v)

  *Split a terrain into subterrains.*

## Static Public Attributes

- static const unsigned int **NumTerrainSubdivisions**

  *DO NOT USE THIS. This is here for ABI compatibilty reasons.*

### 10.77.1 Detailed Description

Rendering a terrain using heightmap information.

### 10.77.2 Constructor & Destructor Documentation

#### 10.77.2.1 gazebo::rendering::Heightmap::Heightmap ( ScenePtr _scene )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | Pointer to the scene that will contain the heightmap |

#### 10.77.2.2 virtual gazebo::rendering::Heightmap::~Heightmap ( ) [virtual]

Destructor.

### 10.77.3 Member Function Documentation

#### 10.77.3.1 bool gazebo::rendering::Heightmap::Flatten ( CameraPtr _camera, math::Vector2i _mousePos, double _outsideRadius, double _insideRadius, double _weight = 0.1 )

Flatten the terrain based on a mouse press.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | **Camera** (p. 186) associated with the mouse press. |
| in | *_mousePos* | Position of the mouse in viewport coordinates. |
| in | *_outsideRadius* | Controls the radius of effect. |
| in | *_insideRadius* | Controls the size of the radius with the maximum effect (value between 0 and 1). |
| in | *_weight* | Controls modification magnitude. |

**Returns**

True if the terrain was modified

#### 10.77.3.2 double gazebo::rendering::Heightmap::GetAvgHeight ( Ogre::Vector3 _pos, double _brushSize )

Get the average height around a point.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | Position in world coordinates. |
| in | *_brushSize* | Controls the radius of effect. |

#### 10.77.3.3 double gazebo::rendering::Heightmap::GetHeight ( double _x, double _y, double _z = 1000 )

Get the height at a location.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | X location |
| in | *_y* | Y location |
| in | *_z* | Z location |

**Returns**

The height at the specified location

#### 10.77.3.4 common::Image gazebo::rendering::Heightmap::GetImage ( ) const

Get the heightmap as an image.

**Returns**

An image that contains the terrain data.

#### 10.77.3.5 Ogre::TerrainGroup::RayResult gazebo::rendering::Heightmap::GetMouseHit ( CameraPtr *_camera,* math::Vector2i *_mousePos* )

Calculate a mouse ray hit on the terrain.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | **Camera** (p. 186) associated with the mouse press. |
| in | *_mousePos* | Position of the mouse in viewport coordinates. |

**Returns**

The result of the mouse ray hit.

#### 10.77.3.6 Ogre::TerrainGroup* gazebo::rendering::Heightmap::GetOgreTerrain ( ) const

Get a pointer to the OGRE terrain group object.

**Returns**

Pointer to the OGRE terrain.

#### 10.77.3.7 unsigned int gazebo::rendering::Heightmap::GetTerrainSubdivisionCount ( ) const

Get the number of subdivision the terrain will be split into.

**Returns**

Number of terrain subdivisions

#### 10.77.3.8 void gazebo::rendering::Heightmap::Load ( )

Load the heightmap.

#### 10.77.3.9 void gazebo::rendering::Heightmap::LoadFromMsg ( ConstVisualPtr & *_msg* )

Load the heightmap from a visual message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | The visual message containing heightmap info |

#### 10.77.3.10 bool gazebo::rendering::Heightmap::Lower ( CameraPtr *_camera*, math::Vector2i *_mousePos*, double *_outsideRadius*, double *_insideRadius*, double *_weight* = 0.1 )

Lower the terrain based on a mouse press.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | **Camera** (p. 186) associated with the mouse press. |
| in | *_mousePos* | Position of the mouse in viewport coordinates. |
| in | *_outsideRadius* | Controls the radius of effect. |
| in | *_insideRadius* | Controls the size of the radius with the maximum effect (value between 0 and 1). |
| in | *_weight* | Controls modification magnitude. |

**Returns**

True if the terrain was modified

#### 10.77.3.11 bool gazebo::rendering::Heightmap::Raise ( CameraPtr *_camera*, math::Vector2i *_mousePos*, double *_outsideRadius*, double *_insideRadius*, double *_weight* = 0.1 )

Raise the terrain based on a mouse press.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | **Camera** (p. 186) associated with the mouse press. |
| in | *_mousePos* | Position of the mouse in viewport coordinates. |
| in | *_outsideRadius* | Controls the radius of effect. |
| in | *_insideRadius* | Controls the size of the radius with the maximum effect (value between 0 and 1). |
| in | *_weight* | Controls modification magnitude. |

**Returns**

True if the terrain was modified

#### 10.77.3.12 void gazebo::rendering::Heightmap::SetWireframe ( bool *_show* )

Set the heightmap to render in wireframe mode.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to render wireframe, false to render solid. |

#### 10.77.3.13 bool gazebo::rendering::Heightmap::Smooth ( CameraPtr *_camera,* math::Vector2i *_mousePos,* double *_outsideRadius,* double *_insideRadius,* double *_weight* = 0.1 )

Smooth the terrain based on a mouse press.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | **Camera** (p. 186) associated with the mouse press. |
| in | *_mousePos* | Position of the mouse in viewport coordinates. |
| in | *_outsideRadius* | Controls the radius of effect. |
| in | *_insideRadius* | Controls the size of the radius with the maximum effect (value between 0 and 1). |
| in | *_weight* | Controls modification magnitude. |

**Returns**

True if the terrain was modified

#### 10.77.3.14 void gazebo::rendering::Heightmap::SplitHeights ( const std::vector< float > & *_heightmap,* int *_n,* std::vector< std::vector< float > > & *_v* )

Split a terrain into subterrains.

**Parameters**

| | | |
|---|---|---|
| in | *_heightmap* | Source vector of floats with the heights. |
| in | *_n* | Number of subterrains. |
| out | *_v* | Destination vector with the subterrains. |

### 10.77.4 Member Data Documentation

#### 10.77.4.1 const unsigned int gazebo::rendering::Heightmap::NumTerrainSubdivisions [static]

DO NOT USE THIS. This is here for ABI compatibilty reasons.

The documentation for this class was generated from the following file:

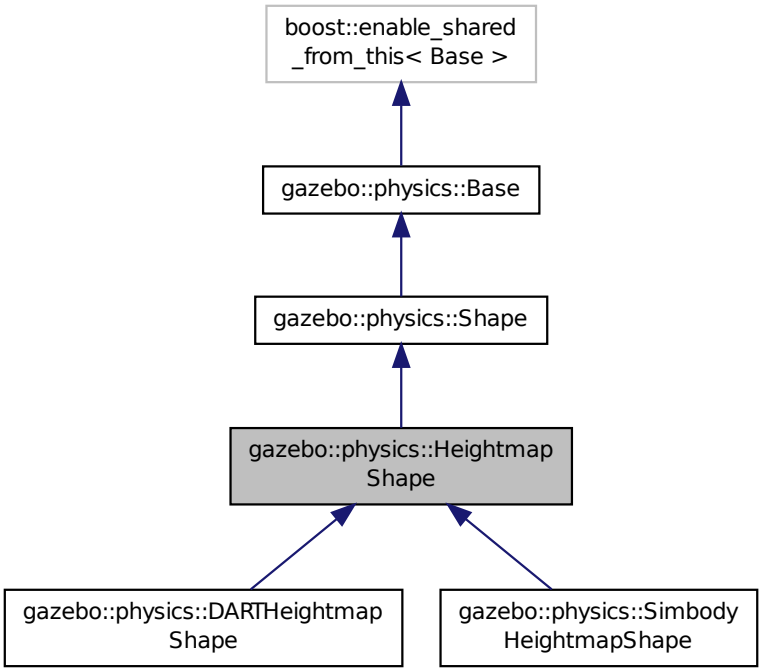
- **Heightmap.hh**

## 10.78 gazebo::physics::HeightmapShape Class Reference

**HeightmapShape** (p. 465) collision shape builds a heightmap from an image.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::HeightmapShape:




## Public Member Functions

- **HeightmapShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**HeightmapShape** ()

  *Destructor.*
- void **FillMsg** (msgs::Geometry &_msg)

  *Fill a geometry message with this shape's data.*
- float **GetHeight** (int _x, int _y) const

  *Get a height at a position.*
- **common::Image GetImage** () const

  *Return an image representation of the heightmap.*
- float **GetMaxHeight** () const

  *Get the maximum height.*
- float **GetMinHeight** () const

  *Get the minimum height.*
- **math::Vector3 GetPos** () const

  *Get the origin in world coordinate frame.*
- **math::Vector3 GetSize** () const

  *Get the size in meters.*

- int **GetSubSampling** () const

  *Get the amount of subsampling.*
- std::string **GetURI** () const

  *Get the URI of the heightmap image.*
- **math::Vector2i GetVertexCount** () const

  *Return the number of vertices, which equals the size of the image used to load the heightmap.*
- virtual void **Init** ()

  *Initialize the heightmap.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the heightmap.*
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *Update the heightmap from a message.*
- virtual void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the heightmap shape.*

## Protected Attributes

- bool **flipY**

  *True to flip the heights along the y direction.*
- std::vector< float > **heights**

  *Lookup table of heights.*
- **common::Image img**

  *Image used to generate the heights.*
- int **subSampling**

  *The amount of subsampling. Default is 2.*
- unsigned int **vertSize**

  *Size of the height lookup table.*

## Additional Inherited Members

### 10.78.1 Detailed Description

**HeightmapShape** (p. 465) collision shape builds a heightmap from an image.

The supplied image must be square with N∗N+1 pixels per side, where N is an integer.

### 10.78.2 Constructor & Destructor Documentation

#### 10.78.2.1 gazebo::physics::HeightmapShape::HeightmapShape ( **CollisionPtr** *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent **Collision** (p. 220) object. |

#### 10.78.2.2 virtual gazebo::physics::HeightmapShape::~HeightmapShape ( ) [virtual]

Destructor.

### 10.78.3 Member Function Documentation

#### 10.78.3.1 void gazebo::physics::HeightmapShape::FillMsg ( msgs::Geometry & *_msg* ) [virtual]

Fill a geometry message with this shape's data.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to fill. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.78.3.2 float gazebo::physics::HeightmapShape::GetHeight ( int *_x*, int *_y* ) const

Get a height at a position.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | X position. |
| in | *_y* | Y position. |

**Returns**

The height at a the specified location.

#### 10.78.3.3 common::Image gazebo::physics::HeightmapShape::GetImage ( ) const

Return an image representation of the heightmap.

**Returns**

Image where white pixels represents the highest locations, and black pixels the lowest.

#### 10.78.3.4 float gazebo::physics::HeightmapShape::GetMaxHeight ( ) const

Get the maximum height.

**Returns**

The maximum height.

#### 10.78.3.5 float gazebo::physics::HeightmapShape::GetMinHeight ( ) const

Get the minimum height.

**Returns**

The minimum height.

#### 10.78.3.6 math::Vector3 gazebo::physics::HeightmapShape::GetPos ( ) const

Get the origin in world coordinate frame.

**Returns**

The origin in world coordinate frame.

#### 10.78.3.7 math::Vector3 gazebo::physics::HeightmapShape::GetSize ( ) const

Get the size in meters.

**Returns**

The size in meters.

#### 10.78.3.8 int gazebo::physics::HeightmapShape::GetSubSampling ( ) const

Get the amount of subsampling.

**Returns**

Amount of subsampling.

#### 10.78.3.9 std::string gazebo::physics::HeightmapShape::GetURI ( ) const

Get the URI of the heightmap image.

**Returns**

The heightmap image URI.

#### 10.78.3.10 math::Vector2i gazebo::physics::HeightmapShape::GetVertexCount ( ) const

Return the number of vertices, which equals the size of the image used to load the heightmap.

**Returns**

**math::Vector2i** (p. 1083), result.x = width, result.y = length/height.

#### 10.78.3.11 virtual void gazebo::physics::HeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 864).

Reimplemented in **gazebo::physics::SimbodyHeightmapShape** (p. 879), and **gazebo::physics::DARTHeightmapShape** (p. 294).

#### 10.78.3.12 virtual void gazebo::physics::HeightmapShape::Load ( sdf::ElementPtr _sdf ) [virtual]

Load the heightmap.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF value to load from. |

Reimplemented from **gazebo::physics::Base** (p. 168).

#### 10.78.3.13 virtual void gazebo::physics::HeightmapShape::ProcessMsg ( const msgs::Geometry & _msg ) [virtual]

Update the heightmap from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to update from. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.78.3.14 virtual void gazebo::physics::HeightmapShape::SetScale ( const math::Vector3 & _scale ) [virtual]

Set the scale of the heightmap shape.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the heightmap shape to. |

Implements **gazebo::physics::Shape** (p. 864).

### 10.78.4 Member Data Documentation

#### 10.78.4.1 bool gazebo::physics::HeightmapShape::flipY [protected]

True to flip the heights along the y direction.

#### 10.78.4.2 std::vector<float> gazebo::physics::HeightmapShape::heights [protected]

Lookup table of heights.

#### 10.78.4.3 common::Image gazebo::physics::HeightmapShape::img [protected]

Image used to generate the heights.

#### 10.78.4.4 int gazebo::physics::HeightmapShape::subSampling [protected]

The amount of subsampling. Default is 2.

#### 10.78.4.5 unsigned int gazebo::physics::HeightmapShape::vertSize [protected]

Size of the height lookup table.

The documentation for this class was generated from the following file:

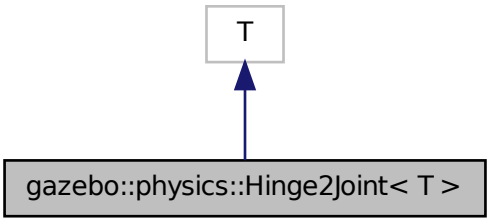
- **HeightmapShape.hh**

## 10.79 gazebo::physics::Hinge2Joint< T > Class Template Reference

A two axis hinge joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Hinge2Joint< T >:




### Public Member Functions

- **Hinge2Joint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**Hinge2Joint** ()

  *Destructor.*
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the joint.*

### 10.79.1 Detailed Description

**template<class T>class gazebo::physics::Hinge2Joint< T >**

A two axis hinge joint.

### 10.79.2 Constructor & Destructor Documentation

#### 10.79.2.1 template<class T> gazebo::physics::Hinge2Joint< T >::Hinge2Joint ( BasePtr _parent ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent link. |

#### 10.79.2.2 template<class T> virtual gazebo::physics::Hinge2Joint< T >::~Hinge2Joint ( ) [inline], [virtual]

Destructor.

### 10.79.3 Member Function Documentation

#### 10.79.3.1 template<class T> virtual unsigned int gazebo::physics::Hinge2Joint< T >::GetAngleCount ( ) const [inline], [virtual]

#### 10.79.3.2 template<class T> virtual void gazebo::physics::Hinge2Joint< T >::Load ( sdf::ElementPtr _sdf ) [inline], [virtual]

Load the joint.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented in **gazebo::physics::SimbodyHinge2Joint** (p. 884), and **gazebo::physics::DARTHinge2Joint** (p. 298).

The documentation for this class was generated from the following file:

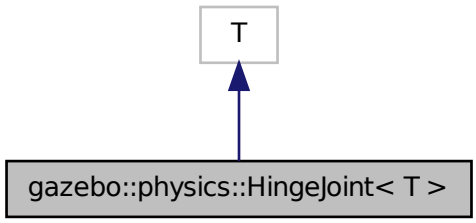
- **Hinge2Joint.hh**

## 10.80 gazebo::physics::HingeJoint< T > Class Template Reference

A single axis hinge joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::HingeJoint< T >:




## Public Member Functions

- **HingeJoint** (**BasePtr** _parent)
  *Constructor.*
- virtual ~**HingeJoint** ()
  *Destructor.*
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
  *Load joint.*

## Protected Member Functions

- virtual void **Init** ()
  *Initialize joint.*

### 10.80.1 Detailed Description

**template<class T>class gazebo::physics::HingeJoint< T >**

A single axis hinge joint.

### 10.80.2 Constructor & Destructor Documentation

#### 10.80.2.1 template<class T> gazebo::physics::HingeJoint< T >::HingeJoint ( BasePtr *_parent* ) [inline]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent link |

#### 10.80.2.2 template<class T> virtual gazebo::physics::HingeJoint< T >::~HingeJoint ( ) [inline], [virtual]

Destructor.

### 10.80.3 Member Function Documentation

#### 10.80.3.1 template<class T> virtual unsigned int gazebo::physics::HingeJoint< T >::GetAngleCount ( ) const [inline], [virtual]

#### 10.80.3.2 template<class T> virtual void gazebo::physics::HingeJoint< T >::Init ( ) [inline], [protected], [virtual]

Initialize joint.

Reimplemented in **gazebo::physics::DARTHingeJoint** (p. 303).

#### 10.80.3.3 template<class T> virtual void gazebo::physics::HingeJoint< T >::Load ( sdf::ElementPtr *_sdf* ) [inline], [virtual]

Load joint.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to SDF element |

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 889), and **gazebo::physics::DARTHingeJoint** (p. 303).

The documentation for this class was generated from the following file:

- **HingeJoint.hh**

## 10.81 gazebo::common::Image Class Reference

Encapsulates an image.

`#include <common/common.hh>`

### Public Types

- enum **PixelFormat** {
  **UNKNOWN_PIXEL_FORMAT** = 0, **L_INT8**, **L_INT16**, **RGB_INT8**,
  **RGBA_INT8**, **BGRA_INT8**, **RGB_INT16**, **RGB_INT32**,
  **BGR_INT8**, **BGR_INT16**, **BGR_INT32**, **R_FLOAT16**,
  **RGB_FLOAT16**, **R_FLOAT32**, **RGB_FLOAT32**, **BAYER_RGGB8**,
  **BAYER_RGGR8**, **BAYER_GBRG8**, **BAYER_GRBG8**, **PIXEL_FORMAT_COUNT** }

  *Pixel formats enumeration.*

### Public Member Functions

- **Image** (const std::string &_filename="")

*Constructor.*

- virtual ∼**Image** ()

  *Destructor.*

- **Color GetAvgColor** ()

  *Get the average color.*

- unsigned int **GetBPP** () const

  *Get the size of one pixel in bits.*

- void **GetData** (unsigned char ∗∗_data, unsigned int &_count) const

  *Get the image as a data array.*

- std::string **GetFilename** () const

  *Get the full filename of the image.*

- unsigned int **GetHeight** () const

  *Get the height.*

- **Color GetMaxColor** ()

  *Get the max color.*

- int **GetPitch** () const
- **Color GetPixel** (unsigned int _x, unsigned int _y)

  *Get a pixel color value.*

- **PixelFormat GetPixelFormat** () const

  *Get the pixel format.*

- void **GetRGBData** (unsigned char ∗∗_data, unsigned int &_count) const

  *Get only the RGB data from the image.*

- unsigned int **GetWidth** () const

  *Get the width.*

- int **Load** (const std::string &_filename)

  *Load an image.*

- void **Rescale** (int _width, int _height)

  *Rescale the image.*

- void **SavePNG** (const std::string &_filename)

  *Save the image in PNG format.*

- void **SetFromData** (const unsigned char ∗_data, unsigned int _width, unsigned int _height, **Image::PixelFormat** _format)

  *Set the image from raw data.*

- bool **Valid** () const

  *Returns whether this is a valid image.*

## Static Public Member Functions

- static **Image::PixelFormat ConvertPixelFormat** (const std::string &_format)

  *Convert a string to a **Image::PixelFormat** (p. 476).*

### 10.81.1 Detailed Description

Encapsulates an image.

### 10.81.2 Member Enumeration Documentation

#### 10.81.2.1 enum gazebo::common::Image::PixelFormat

Pixel formats enumeration.

**Enumerator**

***UNKNOWN_PIXEL_FORMAT***

***L_INT8***

***L_INT16***

***RGB_INT8***

***RGBA_INT8***

***BGRA_INT8***

***RGB_INT16***

***RGB_INT32***

***BGR_INT8***

***BGR_INT16***

***BGR_INT32***

***R_FLOAT16***

***RGB_FLOAT16***

***R_FLOAT32***

***RGB_FLOAT32***

***BAYER_RGGB8***

***BAYER_RGGR8***

***BAYER_GBRG8***

***BAYER_GRBG8***

***PIXEL_FORMAT_COUNT***

### 10.81.3 Constructor & Destructor Documentation

#### 10.81.3.1 gazebo::common::Image::Image ( const std::string & *_filename* = " " ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | the path to the image |

#### 10.81.3.2 virtual gazebo::common::Image::~Image ( ) [virtual]

Destructor.

### 10.81.4 Member Function Documentation

#### 10.81.4.1 static Image::PixelFormat gazebo::common::Image::ConvertPixelFormat ( const std::string & _format ) [static]

Convert a string to a **Image::PixelFormat** (p. 476).

**Parameters**

| | | |
|---|---|---|
| in | *_format* | Pixel format string. |

**See Also**

Image::PixelFormatNames

**Returns**

**Image::PixelFormat** (p. 476)

#### 10.81.4.2 Color gazebo::common::Image::GetAvgColor ( )

Get the average color.

**Returns**

The average color

#### 10.81.4.3 unsigned int gazebo::common::Image::GetBPP ( ) const

Get the size of one pixel in bits.

**Returns**

The BPP of the image

#### 10.81.4.4 void gazebo::common::Image::GetData ( unsigned char ** _data, unsigned int & _count ) const

Get the image as a data array.

**Parameters**

| | | |
|---|---|---|
| out | *_data* | Pointer to a NULL array of char. |
| out | *_count* | The resulting data array size |

#### 10.81.4.5 std::string gazebo::common::Image::GetFilename ( ) const

Get the full filename of the image.

**Returns**

The filename used to load the image

#### 10.81.4.6 unsigned int gazebo::common::Image::GetHeight ( ) const

Get the height.

**Returns**

The image height

#### 10.81.4.7 Color gazebo::common::Image::GetMaxColor ( )

Get the max color.

**Returns**

The max color

#### 10.81.4.8 int gazebo::common::Image::GetPitch ( ) const

**Returns**

The pitch of the image

#### 10.81.4.9 Color gazebo::common::Image::GetPixel ( unsigned int *_x*, unsigned int *_y* )

Get a pixel color value.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | Column location in the image |
| in | *_y* | Row location in the image |

#### 10.81.4.10 PixelFormat gazebo::common::Image::GetPixelFormat ( ) const

Get the pixel format.

**Returns**

PixelFormat

#### 10.81.4.11 void gazebo::common::Image::GetRGBData ( unsigned char ** *_data*, unsigned int & *_count* ) const

Get only the RGB data from the image.

This will drop the alpha channel if one is present.

**Parameters**

| | | |
|---|---|---|
| out | *_data* | Pointer to a NULL array of char. |
| out | *_count* | The resulting data array size |

#### 10.81.4.12 unsigned int gazebo::common::Image::GetWidth ( ) const

Get the width.

**Returns**

The image width

#### 10.81.4.13 int gazebo::common::Image::Load ( const std::string & _filename )

Load an image.

Return 0 on success

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | the path to the image file |

#### 10.81.4.14 void gazebo::common::Image::Rescale ( int _width, int _height )

Rescale the image.

**Parameters**

| | | |
|---|---|---|
| in | *_width* | New image width |
| in | *_height* | New image height |

#### 10.81.4.15 void gazebo::common::Image::SavePNG ( const std::string & _filename )

Save the image in PNG format.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | The name of the saved image |

#### 10.81.4.16 void gazebo::common::Image::SetFromData ( const unsigned char * _data, unsigned int _width, unsigned int _height, Image::PixelFormat _format )

Set the image from raw data.

**Parameters**

| | | |
|---|---|---|
| in | *_data* | Pointer to the raw image data |
| in | *_width* | Width in pixels |
| in | *_height* | Height in pixels |
| in | *_format* | Pixel format of the provided data |

#### 10.81.4.17 bool gazebo::common::Image::Valid ( ) const

Returns whether this is a valid image.

**Returns**

true if image has a bitmap

The documentation for this class was generated from the following file:

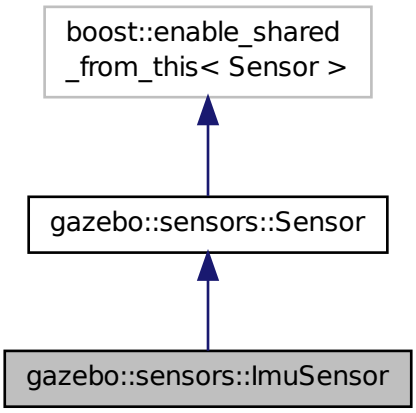
- **Image.hh**

## 10.82 gazebo::sensors::ImuSensor Class Reference

An IMU sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ImuSensor:




### Public Member Functions

- **ImuSensor** ()
  *Constructor.*
- virtual ~**ImuSensor** ()
  *Destructor.*
- **math::Vector3 GetAngularVelocity** () const
  *Returns the angular velocity.*
- msgs::IMU **GetImuMessage** () const
  *Returns the imu message.*

- **math::Vector3 GetLinearAcceleration** () const

  *Returns the imu linear acceleration.*
- **math::Quaternion GetOrientation** () const

  *get orientation of the IMU relative to the reference pose*
- virtual void **Init** ()

  *Initialize the IMU.*
- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*
- void **SetReferencePose** ()

  *Sets the current pose as the IMU reference pose.*

### Protected Member Functions

- virtual void **Fini** ()

  *Finalize the sensor.*
- void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  *Load the sensor with SDF parameters.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*
- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Additional Inherited Members

## 10.82.1 Detailed Description

An IMU sensor.

## 10.82.2 Constructor & Destructor Documentation

#### 10.82.2.1 gazebo::sensors::ImuSensor::ImuSensor ( )

Constructor.

#### 10.82.2.2 virtual gazebo::sensors::ImuSensor::~ImuSensor ( ) [virtual]

Destructor.

## 10.82.3 Member Function Documentation

#### 10.82.3.1 virtual void gazebo::sensors::ImuSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.82.3.2 math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity ( ) const

Returns the angular velocity.

**Returns**

Angular velocity.

#### 10.82.3.3 msgs::IMU gazebo::sensors::ImuSensor::GetImuMessage ( ) const

Returns the imu message.

**Returns**

Imu message.

#### 10.82.3.4 math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration ( ) const

Returns the imu linear acceleration.

**Returns**

Linear acceleration.

#### 10.82.3.5 math::Quaternion gazebo::sensors::ImuSensor::GetOrientation ( ) const

get orientation of the IMU relative to the reference pose

**Returns**

returns the orientation quaternion of the IMU relative to the imu reference pose.

#### 10.82.3.6 virtual void gazebo::sensors::ImuSensor::Init ( ) [virtual]

Initialize the IMU.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.82.3.7 virtual bool gazebo::sensors::ImuSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.82.3.8 void gazebo::sensors::ImuSensor::Load ( const std::string & _worldName, sdf::ElementPtr _sdf ) [protected], [virtual]

Load the sensor with SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF **Sensor** (p. 837) parameters. |
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.82.3.9 virtual void gazebo::sensors::ImuSensor::Load ( const std::string & _worldName ) [protected], [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.82.3.10 void gazebo::sensors::ImuSensor::SetReferencePose ( )

Sets the current pose as the IMU reference pose.

#### 10.82.3.11 virtual void gazebo::sensors::ImuSensor::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

- **ImuSensor.hh**

## 10.83 gazebo::physics::Inertial Class Reference

A class for inertial information about a link.

`#include <physics/physics.hh>`

## Public Member Functions

- **Inertial** ()

  *Default Constructor.*
- **Inertial** (double _mass)

  *Constructor.*
- **Inertial** (const **Inertial** &_inertial)

  *Copy constructor.*
- virtual ~**Inertial** ()

  *Destructor.*
- const **math::Vector3** & **GetCoG** () const

  *Get the center of gravity.*
- **Inertial GetInertial** (const **math::Pose** &_frameOffset) const

  *Get equivalent Inertia values with the **Link** (p. 542) frame offset, while holding the Pose of CoG constant in the world frame.*
- double **GetIXX** () const

  *Get IXX.*
- double **GetIXY** () const

  *Get IXY.*
- double **GetIXZ** () const

  *Get IXZ.*
- double **GetIYY** () const

  *Get IYY.*
- double **GetIYZ** () const

  *Get IXZ.*
- double **GetIZZ** () const

  *Get IZZ.*
- double **GetMass** () const

  *Get the mass.*
- **math::Matrix3 GetMOI** (const **math::Pose** &_pose) const

  *Get the equivalent inertia from a point in local **Link** (p. 542) frame If you specify GetMOI(this->**GetPose()** (p. 488)), you should get back the Moment of Inertia (MOI) exactly as specified in the SDF.*
- **math::Matrix3 GetMOI** () const

  *returns Moments of Inertia as a Matrix3*
- const **math::Pose GetPose** () const

  *Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 542) frame.*
- **math::Vector3 GetPrincipalMoments** () const

  *Get the principal moments of inertia (Ixx, Iyy, Izz).*
- **math::Vector3 GetProductsofInertia** () const

  *Get the products of inertia (Ixy, Ixz, Iyz).*
- void **Load** (sdf::ElementPtr _sdf)

  *Load from SDF values.*
- **Inertial operator+** (const **Inertial** &_inertial) const

  *Addition operator.*
- const **Inertial** & **operator+=** (const **Inertial** &_inertial)

  *Addition equal operator.*
- **Inertial** & **operator=** (const **Inertial** &_inertial)

  *Equal operator.*

- void **ProcessMsg** (const msgs::Inertial &_msg)

  *Update parameters from a message.*
- void **Reset** ()

  *Reset all the mass properties.*
- void **Rotate** (const **math::Quaternion** &_rot)

  *Rotate this mass.*
- void **SetCoG** (double _cx, double _cy, double _cz)

  *Set the center of gravity.*
- void **SetCoG** (const **math::Vector3** &_center)

  *Set the center of gravity.*
- void **SetCoG** (double _cx, double _cy, double _cz, double _rx, double _ry, double _rz)

  *Set the center of gravity and rotation offset of inertial coordinate frame relative to **Link** (p. 542) frame.*
- void **SetCoG** (const **math::Pose** &_c)

  *Set the center of gravity.*
- void **SetInertiaMatrix** (double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double iyz)

  *Set the mass matrix.*
- void **SetIXX** (double _v)

  *Set IXX.*
- void **SetIXY** (double _v)

  *Set IXY.*
- void **SetIXZ** (double _v)

  *Set IXZ.*
- void **SetIYY** (double _v)

  *Set IYY.*
- void **SetIYZ** (double _v)

  *Set IYZ.*
- void **SetIZZ** (double _v)

  *Set IZZ.*
- void **SetMass** (double m)

  *Set the mass.*
- void **SetMOI** (const **math::Matrix3** &_moi)

  *Sets Moments of Inertia (MOI) from a Matrix3.*
- void **UpdateParameters** (sdf::ElementPtr _sdf)

  *update the parameters using new sdf values.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::Inertial** &_inertial)

  *Output operator.*

### 10.83.1 Detailed Description

A class for inertial information about a link.

### 10.83.2 Constructor & Destructor Documentation

#### 10.83.2.1 gazebo::physics::Inertial::Inertial ( )

Default Constructor.

#### 10.83.2.2 gazebo::physics::Inertial::Inertial ( double *_mass* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_mass* | Mass value in kg if using metric. |

#### 10.83.2.3 gazebo::physics::Inertial::Inertial ( const **Inertial** & *_inertial* )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_inertial* | **Inertial** (p. 483) element to copy |

#### 10.83.2.4 virtual gazebo::physics::Inertial::~Inertial ( ) [virtual]

Destructor.

### 10.83.3 Member Function Documentation

#### 10.83.3.1 const math::Vector3& gazebo::physics::Inertial::GetCoG ( ) const [inline]

Get the center of gravity.

**Returns**

The center of gravity.

References gazebo::math::Pose::pos.

#### 10.83.3.2 **Inertial** gazebo::physics::Inertial::GetInertial ( const math::Pose & *_frameOffset* ) const

Get equivalent Inertia values with the **Link** (p. 542) frame offset, while holding the Pose of CoG constant in the world frame.

**Parameters**

| | | |
|---|---|---|
| in | *_frameOffset* | amount to offset the **Link** (p. 542) frame by, this is a transform defined in the **Link** (p. 542) frame. |

**Returns**

**Inertial** (p. 483) parameters with the shifted frame.

#### 10.83.3.3 double gazebo::physics::Inertial::GetIXX ( ) const

Get IXX.

**Returns**

IXX value

#### 10.83.3.4 double gazebo::physics::Inertial::GetIXY ( ) const

Get IXY.

**Returns**

IXY value

#### 10.83.3.5 double gazebo::physics::Inertial::GetIXZ ( ) const

Get IXZ.

**Returns**

IXZ value

#### 10.83.3.6 double gazebo::physics::Inertial::GetIYY ( ) const

Get IYY.

**Returns**

IYY value

#### 10.83.3.7 double gazebo::physics::Inertial::GetIYZ ( ) const

Get IXZ.

**Returns**

IYZ value

#### 10.83.3.8 double gazebo::physics::Inertial::GetIZZ ( ) const

Get IZZ.

**Returns**

IZZ value

#### 10.83.3.9 double gazebo::physics::Inertial::GetMass ( ) const

Get the mass.

#### 10.83.3.10 math::Matrix3 gazebo::physics::Inertial::GetMOI ( const math::Pose & *_pose* ) const

Get the equivalent inertia from a point in local **Link** (p. 542) frame If you specify GetMOI(this->**GetPose()** (p. 488)), you should get back the Moment of Inertia (MOI) exactly as specified in the SDF.

If _pose is different from pose of the **Inertial** (p. 483) block, then the MOI is rotated accordingly, and contributions from changes in MOI location location due to point mass is added to the final MOI.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | location in **Link** (p. 542) local frame |

**Returns**

equivalent inertia at _pose

#### 10.83.3.11 math::Matrix3 gazebo::physics::Inertial::GetMOI ( ) const

returns Moments of Inertia as a Matrix3

**Returns**

Moments of Inertia as a Matrix3

#### 10.83.3.12 const math::Pose gazebo::physics::Inertial::GetPose ( ) const [inline]

Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 542) frame.

**Returns**

The inertial pose.

#### 10.83.3.13 math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments ( ) const

Get the principal moments of inertia (Ixx, Iyy, Izz).

**Returns**

The principal moments.

#### 10.83.3.14 math::Vector3 gazebo::physics::Inertial::GetProductsofInertia ( ) const

Get the products of inertia (Ixy, Ixz, Iyz).

**Returns**

The products of inertia.

#### 10.83.3.15 void gazebo::physics::Inertial::Load ( sdf::ElementPtr _sdf )

Load from SDF values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF value to load from. |

#### 10.83.3.16 Inertial gazebo::physics::Inertial::operator+ ( const Inertial & _inertial ) const

Addition operator.

Assuming both CG and Moment of Inertia (MOI) are defined in the same reference **Link** (p. 542) frame. New CG is computed from masses and perspective offsets, and both MOI contributions relocated to the new cog.

**Parameters**

| | | |
|---|---|---|
| in | *_inertial* | **Inertial** (p. 483) to add. |

**Returns**

The result of the addition.

#### 10.83.3.17 const Inertial& gazebo::physics::Inertial::operator+= ( const Inertial & _inertial )

Addition equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_inertial* | **Inertial** (p. 483) to add. |

**Returns**

Reference to this object.

#### 10.83.3.18 Inertial& gazebo::physics::Inertial::operator= ( const Inertial & _inertial )

Equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_inertial* | **Inertial** (p. 483) to copy. |

**Returns**

Reference to this object.

#### 10.83.3.19 void gazebo::physics::Inertial::ProcessMsg ( const msgs::Inertial & _msg )

Update parameters from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to read |

#### 10.83.3.20 void gazebo::physics::Inertial::Reset ( )

Reset all the mass properties.

#### 10.83.3.21 void gazebo::physics::Inertial::Rotate ( const math::Quaternion & *_rot* )

Rotate this mass.

**Parameters**

| | | |
|---|---|---|
| in | *_rot* | Rotation amount. |

#### 10.83.3.22 void gazebo::physics::Inertial::SetCoG ( double *_cx*, double *_cy*, double *_cz* )

Set the center of gravity.

**Parameters**

| | | |
|---|---|---|
| in | *_cx* | X position. |
| in | *_cy* | Y position. |
| in | *_cz* | Z position. |

#### 10.83.3.23 void gazebo::physics::Inertial::SetCoG ( const math::Vector3 & *_center* )

Set the center of gravity.

**Parameters**

| | | |
|---|---|---|
| in | *_center* | Center of the gravity. |

#### 10.83.3.24 void gazebo::physics::Inertial::SetCoG ( double *_cx*, double *_cy*, double *_cz*, double *_rx*, double *_ry*, double *_rz* )

Set the center of gravity and rotation offset of inertial coordinate frame relative to **Link** (p. 542) frame.

**Parameters**

| | | |
|---|---|---|
| in | *_cx* | Center offset in x-direction in **Link** (p. 542) frame |
| in | *_cy* | Center offset in y-direction in **Link** (p. 542) frame |
| in | *_cz* | Center offset in z-direction in **Link** (p. 542) frame |
| in | *_rx* | Roll angle offset of inertial coordinate frame. |
| in | *_ry* | Pitch angle offset of inertial coordinate frame. |
| in | *_rz* | Yaw angle offset of inertial coordinate frame. |

#### 10.83.3.25 void gazebo::physics::Inertial::SetCoG ( const math::Pose & _c )

Set the center of gravity.

**Parameters**

| | | |
|---|---|---|
| in | _c | Transform to center of gravity. |

#### 10.83.3.26 void gazebo::physics::Inertial::SetInertiaMatrix ( double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double iyz )

Set the mass matrix.

**Parameters**

| | | |
|---|---|---|
| in | _ixx | X second moment of inertia (MOI) about x axis. |
| in | _iyy | Y second moment of inertia about y axis. |
| in | _izz | Z second moment of inertia about z axis. |
| in | _ixy | XY inertia. |
| in | _ixz | XZ inertia. |
| in | _iyz | YZ inertia. |

#### 10.83.3.27 void gazebo::physics::Inertial::SetIXX ( double _v )

Set IXX.

**Parameters**

| | | |
|---|---|---|
| in | _v | IXX value |

#### 10.83.3.28 void gazebo::physics::Inertial::SetIXY ( double _v )

Set IXY.

**Parameters**

| | | |
|---|---|---|
| in | _v | IXY value |

#### 10.83.3.29 void gazebo::physics::Inertial::SetIXZ ( double _v )

Set IXZ.

**Parameters**

| | | |
|---|---|---|
| in | _v | IXZ value |

#### 10.83.3.30 void gazebo::physics::Inertial::SetIYY ( double _v )

Set IYY.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | IYY value |

#### 10.83.3.31 void gazebo::physics::Inertial::SetIYZ ( double *_v* )

Set IYZ.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | IXX value |

#### 10.83.3.32 void gazebo::physics::Inertial::SetIZZ ( double *_v* )

Set IZZ.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | IZZ value |

#### 10.83.3.33 void gazebo::physics::Inertial::SetMass ( double *m* )

Set the mass.

#### 10.83.3.34 void gazebo::physics::Inertial::SetMOI ( const **math::Matrix3** & *_moi* )

Sets Moments of Inertia (MOI) from a Matrix3.

**Parameters**

| | | |
|---|---|---|
| in | *Moments* | of Inertia as a Matrix3 |

#### 10.83.3.35 void gazebo::physics::Inertial::UpdateParameters ( sdf::ElementPtr *_sdf* )

update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Update values from. |

### 10.83.4 Friends And Related Function Documentation

#### 10.83.4.1 std::ostream& operator<< ( std::ostream & *_out,* const **gazebo::physics::Inertial** & *_inertial* ) [friend]

Output operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | Output stream. |
| in | *_inertial* | **Inertial** (p. 483) object to output. |

The documentation for this class was generated from the following file:

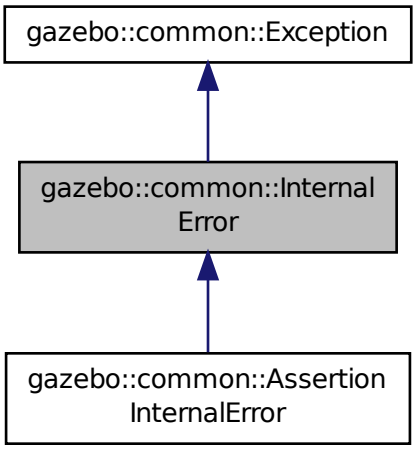
- **Inertial.hh**

## 10.84 gazebo::common::InternalError Class Reference

Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::InternalError:




### Public Member Functions

- **InternalError** ()

  *Constructor.*

- **InternalError** (const char *_file, int _line, const std::string &_msg)

  *Default constructor.*

- virtual ~**InternalError** ()

  *Destructor.*

### 10.84.1 Detailed Description

Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.

### 10.84.2 Constructor & Destructor Documentation

#### 10.84.2.1 gazebo::common::InternalError::InternalError ( )

Constructor.

#### 10.84.2.2 gazebo::common::InternalError::InternalError ( const char * *_file*, int *_line*, const std::string & *_msg* )

Default constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_file* | File name |
| in | *_line* | Line number where the error occurred |
| in | *_msg* | Error message |

#### 10.84.2.3 virtual gazebo::common::InternalError::~InternalError ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Exception.hh**

## 10.85 gazebo::transport::IOManager Class Reference

Manages boost::asio IO.

`#include <transport/transport.hh>`

### Public Member Functions

- **IOManager** ()
  *Constructor.*
- ~**IOManager** ()
  *Destructor.*
- void **DecCount** ()
  *Decrement the event count by 1.*
- unsigned int **GetCount** () const
  *Get the event count.*
- boost::asio::io_service & **GetIO** ()
  *Get handle to boost::asio IO service.*
- void **IncCount** ()

*Increment the event count by 1.*

- void **Stop** ()

  *Stop the IO service.*

### 10.85.1 Detailed Description

Manages boost::asio IO.

### 10.85.2 Constructor & Destructor Documentation

#### 10.85.2.1 gazebo::transport::IOManager::IOManager ( )

Constructor.

#### 10.85.2.2 gazebo::transport::IOManager::~IOManager ( )

Destructor.

### 10.85.3 Member Function Documentation

#### 10.85.3.1 void gazebo::transport::IOManager::DecCount ( )

Decrement the event count by 1.

#### 10.85.3.2 unsigned int gazebo::transport::IOManager::GetCount ( ) const

Get the event count.

**Returns**

The event count

#### 10.85.3.3 boost::asio::io_service& gazebo::transport::IOManager::GetIO ( )

Get handle to boost::asio IO service.

**Returns**

Handle to boost::asio IO service

#### 10.85.3.4 void gazebo::transport::IOManager::IncCount ( )

Increment the event count by 1.

#### 10.85.3.5 void gazebo::transport::IOManager::Stop ( )

Stop the IO service.

The documentation for this class was generated from the following file:

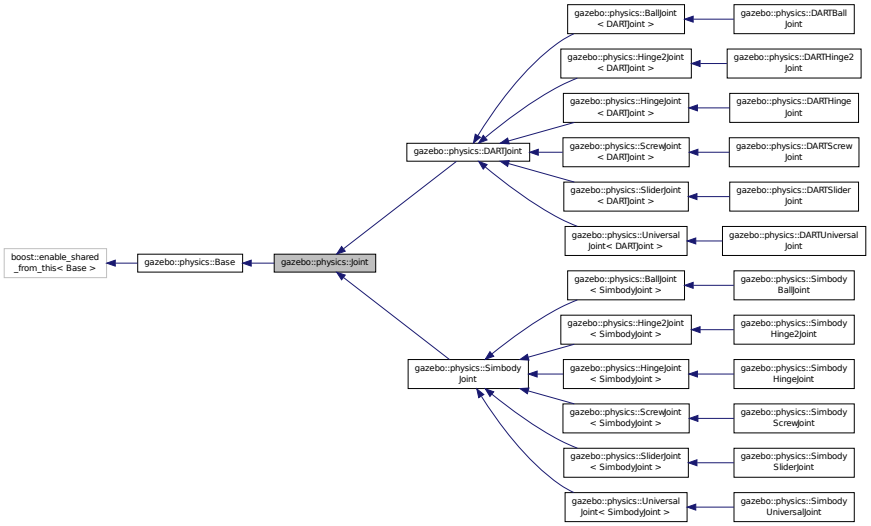
- **IOManager.hh**

## 10.86 gazebo::physics::Joint Class Reference

**Base** (p. 159) class for all joints.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Joint:




### Public Types

- enum **Attribute** {
  **FUDGE_FACTOR**, **SUSPENSION_ERP**, **SUSPENSION_CFM**, **STOP_ERP**,
  **STOP_CFM**, **ERP**, **CFM**, **FMAX**,
  **VEL**, **HI_STOP**, **LO_STOP** }

  ***Joint*** *(p. 496) attribute types.*

### Public Member Functions

- **Joint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**Joint** ()

  *Destructor.*
- virtual void **ApplyDamping** ()

*Callback to apply damping force to joint.*

- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const =0

  *Determines of the two bodies are connected by a joint.*
- virtual void **Attach** (**LinkPtr** _parent, **LinkPtr** _child)

  *Attach the two bodies with this joint.*
- virtual void **CacheForceTorque** ()

  *Cache **Joint** (p. 496) Force Torque Values if necessary for physics engine.*
- double **CheckAndTruncateForce** (int _index, double _effort)

  *check if the force against velocityLimit and effortLimit, truncate if necessary.*
- template<typename T >
  **event::ConnectionPtr ConnectJointUpdate** (T _subscriber)

  *Connect a boost::slot the the joint update signal.*
- virtual void **Detach** ()

  *Detach this joint from all links.*
- void **DisconnectJointUpdate** (**event::ConnectionPtr** &_conn)

  *Disconnect a boost::slot the the joint update signal.*
- void **FillMsg** (msgs::Joint &_msg)

  *Fill a joint message.*
- virtual **math::Vector3 GetAnchor** (int _index) const =0

  *Get the anchor point.*
- **math::Angle GetAngle** (int _index) const

  *Get the angle of rotation of an axis(index)*
- virtual unsigned int **GetAngleCount** () const =0

  *Get the angle count.*
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index)=0

  *Get a non-generic parameter for the joint.*
- **LinkPtr GetChild** () const

  *Get the child link.*
- double **GetDamping** (int _index)

  *Returns the current joint damping coefficient.*
- double **GetDampingCoefficient** () const

  *Get damping coefficient of this joint.*
- virtual double **GetEffortLimit** (int _index)

  *Get the effort limit on axis(index).*
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)=0

  *get internal force and torque values at a joint.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const =0

  *Get the axis of rotation in global cooridnate frame.*
- virtual **math::Angle GetHighStop** (int _index)=0

  *Get the high stop of an axis(index).*
- double **GetInertiaRatio** (unsigned int _index) const

  *Accessor to inertia ratio across this joint.*
- **math::Pose GetInitialAnchorPose** ()

  *Get initial Anchor Pose specified by model <joint><pose>...</pose></joint>*
- virtual **LinkPtr GetJointLink** (int _index) const =0

  *Get the link to which the joint is attached according the _index.*

- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const =0

  *Get the forces applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0

  *Get the torque applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).*
- **math::Vector3 GetLocalAxis** (int _index) const

  *Get the axis of rotation.*
- **math::Angle GetLowerLimit** (unsigned int _index) const

  *: get the joint upper limit (replaces GetLowStop and GetHighStop)*
- virtual **math::Angle GetLowStop** (int _index)=0

  *Get the low stop of an axis(index).*
- virtual double **GetMaxForce** (int _index)=0

  *Get the max allowed force of an axis(index).*
- **LinkPtr GetParent** () const

  *Get the parent link.*
- **math::Angle GetUpperLimit** (unsigned int _index) const

  *: get the joint lower limit (replacee GetLowStop and GetHighStop)*
- virtual double **GetVelocity** (int _index) const =0

  *Get the rotation rate of an axis(index)*
- virtual double **GetVelocityLimit** (int _index)

  *Get the velocity limit on axis(index).*
- virtual void **Init** ()

  *Initialize a joint.*
- void **Load** (**LinkPtr** _parent, **LinkPtr** _child, const **math::Pose** &_pose)

  *Set pose, parent and child links of a **physics::Joint** (p. 496).*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint** (p. 496) from a SDF sdf::Element.*
- virtual void **Reset** ()

  *Reset the joint.*
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)=0

  *Set the anchor point.*
- void **SetAngle** (int _index, **math::Angle** _angle)

  *If the **Joint** (p. 496) is static, Gazebo stores the state of this **Joint** (p. 496) as a scalar inside the **Joint** (p. 496) class, so this call will NOT move the joint dynamically for a static **Model** (p. 624).*
- virtual void **SetAttribute** (const std::string &_key, int _index, const boost::any &_value)=0

  *Set a non-generic parameter for the joint.*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)=0

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, double _damping)=0

  *Set the joint damping.*
- virtual void **SetEffortLimit** (unsigned int _index, double _effort)

  *Set the effort limit on a joint axis.*
- virtual void **SetForce** (int _index, double _effort)=0

  *Set the force applied to this **physics::Joint** (p. 496).*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*

- virtual void **SetMaxForce** (int _index, double _force)=0

  *Set the max allowed force of an axis(index).*
- void **SetModel** (**ModelPtr** _model)

  *Set the model this joint belongs too.*
- virtual void **SetProvideFeedback** (bool _enable)

  *Set whether the joint should generate feedback.*
- void **SetState** (const **JointState** &_state)

  *Set the joint state.*
- virtual void **SetVelocity** (int _index, double _vel)=0

  *Set the velocity of an axis(index).*
- void **Update** ()

  *Update the joint.*
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  *Update the parameters using new sdf values.*

## Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const =0

  *Get the angle of an axis helper function.*

## Protected Attributes

- **LinkPtr anchorLink**

  *Anchor link.*
- **math::Vector3 anchorPos**

  *Anchor pose.*
- **math::Pose anchorPose**

  *Anchor pose specified in SDF <joint><pose> tag.*
- **gazebo::event::ConnectionPtr applyDamping**

  *apply damping for adding viscous damping forces on updates*
- **LinkPtr childLink**

  *The first link this joint connects to.*
- double **dampingCoefficient**

  *joint dampingCoefficient*
- double **effortLimit** [2]

  *Store **Joint** (p. 496) effort limit as specified in SDF.*
- double **inertiaRatio** [2]

  *Store **Joint** (p. 496) inertia ratio.*
- **math::Angle lowerLimit** [2]

  *Store **Joint** (p. 496) position lower limit as specified in SDF.*
- **ModelPtr model**

  *Pointer to the parent model.*
- **LinkPtr parentLink**

  *The second link this joint connects to.*
- bool **provideFeedback**

  *Provide Feedback data for contact forces.*

- **math::Angle upperLimit** [2]

  *Store **Joint** (p. 496) position upper limit as specified in SDF.*
- bool **useCFMDamping**

  *option to use CFM damping*
- double **velocityLimit** [2]

  *Store **Joint** (p. 496) velocity limit as specified in SDF.*
- **JointWrench wrench**

  *Cache **Joint** (p. 496) force torque values in case physics engine clears them at the end of update step.*

### 10.86.1 Detailed Description

**Base** (p. 159) class for all joints.

### 10.86.2 Member Enumeration Documentation

#### 10.86.2.1 enum gazebo::physics::Joint::Attribute

**Joint** (p. 496) attribute types.

**Enumerator**

***FUDGE_FACTOR*** Fudge factor.

***SUSPENSION_ERP*** Suspension error reduction parameter.

***SUSPENSION_CFM*** Suspension constraint force mixing.

***STOP_ERP*** Stop limit error reduction parameter.

***STOP_CFM*** Stop limit constraint force mixing.

***ERP*** Error reduction parameter.

***CFM*** Constraint force mixing.

***FMAX*** Maximum force.

***VEL*** Velocity.

***HI_STOP*** High stop angle.

***LO_STOP*** Low stop angle.

### 10.86.3 Constructor & Destructor Documentation

#### 10.86.3.1 gazebo::physics::Joint::Joint ( BasePtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | ***Joint** (p. 496)* | parent |

#### 10.86.3.2 virtual gazebo::physics::Joint::~Joint ( ) [virtual]

Destructor.

### 10.86.4 Member Function Documentation

#### 10.86.4.1 virtual void gazebo::physics::Joint::ApplyDamping ( ) [virtual]

Callback to apply damping force to joint.

Reimplemented in **gazebo::physics::DARTJoint** (p. 307).

#### 10.86.4.2 virtual bool gazebo::physics::Joint::AreConnected ( LinkPtr *_one,* LinkPtr *_two* ) const [pure virtual]

Determines of the two bodies are connected by a joint.

**Parameters**

| | | |
|---|---|---|
| in | *_one* | First link. |
| in | *_two* | Second link. |

**Returns**

True if the two links are connected by a joint.

Implemented in **gazebo::physics::DARTJoint** (p. 307), and **gazebo::physics::SimbodyJoint** (p. 894).

#### 10.86.4.3 virtual void gazebo::physics::Joint::Attach ( LinkPtr *_parent,* LinkPtr *_child* ) [virtual]

Attach the two bodies with this joint.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent link. |
| in | *_child* | Child link. |

Reimplemented in **gazebo::physics::DARTJoint** (p. 307).

#### 10.86.4.4 virtual void gazebo::physics::Joint::CacheForceTorque ( ) [inline], [virtual]

Cache **Joint** (p. 496) Force Torque Values if necessary for physics engine.

Reimplemented in **gazebo::physics::SimbodyJoint** (p. 894).

#### 10.86.4.5 double gazebo::physics::Joint::CheckAndTruncateForce ( int *_index,* double *_effort* )

check if the force against velocityLimit and effortLimit, truncate if necessary.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_effort* | Force value. |

**Returns**

truncated effort

#### 10.86.4.6 template<typename T > event::ConnectionPtr gazebo::physics::Joint::ConnectJointUpdate ( T _subscriber ) [inline]

Connect a boost::slot the the joint update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback for the connection. |

**Returns**

Connection pointer, which must be kept in scope.

References gazebo::event::EventT< T >::Connect().

#### 10.86.4.7 virtual void gazebo::physics::Joint::Detach ( ) [virtual]

Detach this joint from all links.

Reimplemented in **gazebo::physics::DARTJoint** (p. 308), and **gazebo::physics::SimbodyJoint** (p. 894).

#### 10.86.4.8 void gazebo::physics::Joint::DisconnectJointUpdate ( event::ConnectionPtr & _conn ) [inline]

Disconnect a boost::slot the the joint update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | Connection to disconnect. |

References gazebo::event::EventT< T >::Disconnect().

#### 10.86.4.9 void gazebo::physics::Joint::FillMsg ( msgs::Joint & _msg )

Fill a joint message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message to fill with this joint's properties. |

#### 10.86.4.10 virtual math::Vector3 gazebo::physics::Joint::GetAnchor ( int _index ) const [pure virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Anchor value for the axis.

Implemented in **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), **gazebo::physics::SimbodyJoint** (p. 894), **gazebo::physics::SliderJoint< DARTJoint >** (p. 974), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 974), **gazebo::physics::SimbodyHinge2Joint** (p. 882), **gazebo::physics::SimbodyUniversalJoint** (p. 947), **gazebo::physics::SimbodyBallJoint** (p. 868), **gazebo::physics::DARTHinge2Joint** (p. 297), **gazebo::physics::DARTHingeJoint** (p. 302), **gazebo::physics::DARTBallJoint** (p. 282), **gazebo::physics::DARTSliderJoint** (p. 347), and **gazebo::physics::DARTUniversalJoint** (p. 355).

#### 10.86.4.11 math::Angle gazebo::physics::Joint::GetAngle ( int *_index* ) const

Get the angle of rotation of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

#### 10.86.4.12 virtual unsigned int gazebo::physics::Joint::GetAngleCount ( ) const [pure virtual]

Get the angle count.

**Returns**

The number of DOF for the joint.

Implemented in **gazebo::physics::DARTJoint** (p. 308), **gazebo::physics::BallJoint< DARTJoint >** (p. 158), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 158), **gazebo::physics::SliderJoint< DARTJoint >** (p. 975), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 975), **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 472), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 472), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1065), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1065), **gazebo::physics::HingeJoint< DARTJoint >** (p. 474), and **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 474).

#### 10.86.4.13 virtual math::Angle gazebo::physics::Joint::GetAngleImpl ( int *_index* ) const [protected], [pure virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implemented in **gazebo::physics::SimbodyHinge2Joint** (p. 882), **gazebo::physics::SimbodyUniversalJoint** (p. 947), **gazebo::physics::SimbodyScrewJoint** (p. 932), **gazebo::physics::SimbodyHingeJoint** (p. 888), **gazebo::physics::SimbodySliderJoint** (p. 939), **gazebo::physics::SimbodyBallJoint** (p. 868), **gazebo::physics::DARTBallJoint** (p. 282), **gazebo::physics::DARTHinge2Joint** (p. 297), **gazebo::physics::DARTHingeJoint** (p. 302), **gazebo::physics::DARTScrewJoint** (p. 342), **gazebo::physics::DARTSliderJoint** (p. 347), and **gazebo::physics::DARTUniversalJoint** (p. 355).

#### 10.86.4.14 virtual double gazebo::physics::Joint::GetAttribute ( const std::string & *_key*, unsigned int *_index* ) [pure virtual]

Get a non-generic parameter for the joint.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | String key. |
| in | *_index* | Index of the axis. |
| in | *_value* | Value of the attribute. |

Implemented in **gazebo::physics::DARTJoint** (p. 308), and **gazebo::physics::SimbodyJoint** (p. 895).

#### 10.86.4.15 LinkPtr gazebo::physics::Joint::GetChild ( ) const

Get the child link.

**Returns**

Pointer to the child link.

#### 10.86.4.16 double gazebo::physics::Joint::GetDamping ( int *_index* )

Returns the current joint damping coefficient.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get, currently ignored, to be implemented. |

**Returns**

**Joint** (p. 496) viscous damping coefficient for this joint.

#### 10.86.4.17 double gazebo::physics::Joint::GetDampingCoefficient ( ) const

Get damping coefficient of this joint.

**Returns**

viscous joint damping coefficient

#### 10.86.4.18 virtual double gazebo::physics::Joint::GetEffortLimit ( int *_index* ) [virtual]

Get the effort limit on axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of axis, where 0=first axis and 1=second axis |

**Returns**

Effort limit specified in SDF

#### 10.86.4.19 virtual double gazebo::physics::Joint::GetForce ( unsigned int *_index* ) [virtual]

**Todo** : not yet implemented. Get external forces applied at this **Joint** (p. 496). Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The force applied to an axis.

Reimplemented in **gazebo::physics::DARTJoint** (p. 308), and **gazebo::physics::SimbodyJoint** (p. 895).

#### 10.86.4.20 virtual JointWrench gazebo::physics::Joint::GetForceTorque ( unsigned int *_index* ) [pure virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 529) data structure. Where **JointWrench.body1Force** (p. 530) contains the force applied by the parent **Link** (p. 542) on the **Joint** (p. 496) specified in the parent **Link** (p. 542) frame, and **JointWrench.body2Force** (p. 531) contains the force applied by the child **Link** (p. 542) on the **Joint** (p. 496) specified in the child **Link** (p. 542) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 496) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 542) frames.

Note that for ODE you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Not used right now |

**Returns**

The force and torque at the joint, see above for details on conventions.

Implemented in **gazebo::physics::SimbodyJoint** (p. 895), and **gazebo::physics::DARTJoint** (p. 309).

#### 10.86.4.21 virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis ( int *_index* ) const [pure virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implemented in **gazebo::physics::SimbodyHinge2Joint** (p. 882), **gazebo::physics::SimbodyUniversalJoint** (p. 947), **gazebo::physics::SimbodyScrewJoint** (p. 932), **gazebo::physics::SimbodyBallJoint** (p. 868), **gazebo::physics::SimbodyHingeJoint** (p. 888), **gazebo::physics::SimbodySliderJoint** (p. 939), **gazebo::physics::DARTHinge2Joint** (p. 297), **gazebo::physics::DARTHingeJoint** (p. 302), **gazebo::physics::DARTBallJoint** (p. 282), **gazebo::physics::DARTSliderJoint** (p. 347), **gazebo::physics::DARTUniversalJoint** (p. 355), and **gazebo::physics::DARTScrewJoint** (p. 342).

#### 10.86.4.22 virtual math::Angle gazebo::physics::Joint::GetHighStop ( int *_index* ) [pure virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the high stop value.

Implemented in **gazebo::physics::SimbodyHinge2Joint** (p. 882), **gazebo::physics::SimbodyUniversalJoint** (p. 948), **gazebo::physics::DARTJoint** (p. 309), **gazebo::physics::BallJoint< DARTJoint >** (p. 158), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 158), **gazebo::physics::SimbodyHingeJoint** (p. 888), **gazebo::physics::SimbodySliderJoint** (p. 939), and **gazebo::physics::SimbodyScrewJoint** (p. 932).

#### 10.86.4.23 double gazebo::physics::Joint::GetInertiaRatio ( unsigned int *_index* ) const

Accessor to inertia ratio across this joint.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | **Joint** (p. 496) axis index. |

**Returns**

returns the inertia ratio across specified joint axis.

#### 10.86.4.24 math::Pose gazebo::physics::Joint::GetInitialAnchorPose ( )

Get initial Anchor Pose specified by model <joint><pose>...</pose></joint>

**Returns**

**Joint::anchorPose** (p. 515), initial joint anchor pose.

#### 10.86.4.25 virtual LinkPtr gazebo::physics::Joint::GetJointLink ( int *_index* ) const [pure virtual]

Get the link to which the joint is attached according the _index.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the link to retreive. |

**Returns**

Pointer to the request link. NULL if the index was invalid.

Implemented in **gazebo::physics::DARTJoint** (p. 310), and **gazebo::physics::SimbodyJoint** (p. 896).

#### 10.86.4.26 virtual math::Vector3 gazebo::physics::Joint::GetLinkForce ( unsigned int *_index* ) const [pure virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of the link(0 or 1). |

**Returns**

Force applied to the link.

Implemented in **gazebo::physics::DARTJoint** (p. 310), and **gazebo::physics::SimbodyJoint** (p. 896).

#### 10.86.4.27 virtual math::Vector3 gazebo::physics::Joint::GetLinkTorque ( unsigned int *_index* ) const [pure virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).

Note that the unit of torque should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of the link(0 or 1) |

**Returns**

Torque applied to the link.

Implemented in **gazebo::physics::DARTJoint** (p. 310), and **gazebo::physics::SimbodyJoint** (p. 896).

#### 10.86.4.28 math::Vector3 gazebo::physics::Joint::GetLocalAxis ( int *_index* ) const

Get the axis of rotation.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

#### 10.86.4.29 math::Angle gazebo::physics::Joint::GetLowerLimit ( unsigned int *_index* ) const

: get the joint upper limit (replaces GetLowStop and GetHighStop)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Upper limit of the axis.

#### 10.86.4.30 virtual math::Angle gazebo::physics::Joint::GetLowStop ( int *_index* ) [pure virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the low stop value.

Implemented in **gazebo::physics::SimbodyHinge2Joint** (p. 883), **gazebo::physics::SimbodyUniversalJoint** (p. 948), **gazebo::physics::DARTJoint** (p. 310), **gazebo::physics::BallJoint< DARTJoint >** (p. 158), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 158), **gazebo::physics::SimbodyHingeJoint** (p. 889), **gazebo::physics::SimbodySliderJoint** (p. 939), and **gazebo::physics::SimbodyScrewJoint** (p. 933).

#### 10.86.4.31 virtual double gazebo::physics::Joint::GetMaxForce ( int *_index* ) [pure virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 933), **gazebo::physics::SimbodyHinge2Joint** (p. 883), **gazebo::physics::SimbodyUniversalJoint** (p. 948), **gazebo::physics::DARTHinge2Joint** (p. 297), **gazebo::physics::DARTHingeJoint** (p. 302), **gazebo::physics::DARTScrewJoint** (p. 342), **gazebo::physics::DARTSliderJoint** (p. 348), **gazebo::physics::DARTUniversalJoint** (p. 355), **gazebo::physics::SimbodyBallJoint** (p. 868), **gazebo::physics::SimbodyHingeJoint** (p. 889), **gazebo::physics::SimbodySliderJoint** (p. 940), and **gazebo::physics::DARTBallJoint** (p. 282).

#### 10.86.4.32 LinkPtr gazebo::physics::Joint::GetParent ( ) const

Get the parent link.

**Returns**

Pointer to the parent link.

#### 10.86.4.33 math::Angle gazebo::physics::Joint::GetUpperLimit ( unsigned int *_index* ) const

: get the joint lower limit (replacee GetLowStop and GetHighStop)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Upper limit of the axis.

#### 10.86.4.34 virtual double gazebo::physics::Joint::GetVelocity ( int *_index* ) const [pure virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 933), **gazebo::physics::SimbodyUniversalJoint** (p. 948), **gazebo::physics::SimbodyHinge2Joint** (p. 883), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::SimbodyBallJoint** (p. 869), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTScrewJoint** (p. 343), **gazebo::physics::DARTSliderJoint** (p. 348), **gazebo::physics::SimbodyHingeJoint** (p. 889), **gazebo::physics::DARTUniversalJoint** (p. 356), **gazebo::physics::DARTBallJoint** (p. 283), and **gazebo::physics::SimbodySliderJoint** (p. 940).

#### 10.86.4.35 virtual double gazebo::physics::Joint::GetVelocityLimit ( int _index ) [virtual]

Get the velocity limit on axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of axis, where 0=first axis and 1=second axis |

**Returns**

Velocity limit specified in SDF

#### 10.86.4.36 virtual void gazebo::physics::Joint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Base** (p. 167).

Reimplemented in **gazebo::physics::HingeJoint< DARTJoint >** (p. 474), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 474), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyScrewJoint** (p. 934), **gazebo::physics::SimbodyBallJoint** (p. 869), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::DARTJoint** (p. 311), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::DARTBallJoint** (p. 283), **gazebo::physics::DARTScrewJoint** (p. 343), **gazebo::physics::DARTSliderJoint** (p. 348), and **gazebo::physics::DARTUniversalJoint** (p. 356).

#### 10.86.4.37 void gazebo::physics::Joint::Load ( LinkPtr _parent, LinkPtr _child, const math::Pose & _pose )

Set pose, parent and child links of a **physics::Joint** (p. 496).

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent link. |
| in | *_child* | Child link. |
| in | *_pose* | Pose containing **Joint** (p. 496) Anchor offset from child link. |

#### 10.86.4.38 virtual void gazebo::physics::Joint::Load ( sdf::ElementPtr _sdf ) [virtual]

Load **physics::Joint** (p. 496) from a SDF sdf::Element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::Base** (p. 168).

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 940), **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 472), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 472), **gazebo::physics::BallJoint< DARTJoint >** (p. 158), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 158), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1065), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1065), **gazebo::physics::HingeJoint< DARTJoint >** (p. 474), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 474), **gazebo::physics::SliderJoint< DARTJoint >** (p. 975), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 975), **gazebo::physics::SimbodyHingeJoint** (p. 889), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyUniversalJoint** (p. 949),

**gazebo::physics::SimbodyJoint** (p. 897), **gazebo::physics::SimbodyScrewJoint** (p. 934), **gazebo::physics::DARTJoint** (p. 311), **gazebo::physics::SimbodyBallJoint** (p. 869), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::DARTBallJoint** (p. 283), **gazebo::physics::DARTScrewJoint** (p. 343), **gazebo::physics::DARTSliderJoint** (p. 348), and **gazebo::physics::DARTUniversalJoint** (p. 356).

#### 10.86.4.39 virtual void gazebo::physics::Joint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Base** (p. 169).

Reimplemented in **gazebo::physics::DARTJoint** (p. 311), and **gazebo::physics::SimbodyJoint** (p. 897).

#### 10.86.4.40 virtual void gazebo::physics::Joint::SetAnchor ( int *_index*, const math::Vector3 & *_anchor* ) [pure virtual]

Set the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Indx of the axis. |
| in | *_anchor* | Anchor value. |

Implemented in **gazebo::physics::ScrewJoint< DARTJoint >** (p. 834), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), **gazebo::physics::DARTJoint** (p. 311), **gazebo::physics::SimbodyJoint** (p. 897), **gazebo::physics::SliderJoint< DARTJoint >** (p. 975), and **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 975).

#### 10.86.4.41 void gazebo::physics::Joint::SetAngle ( int *_index*, math::Angle *_angle* )

If the **Joint** (p. 496) is static, Gazebo stores the state of this **Joint** (p. 496) as a scalar inside the **Joint** (p. 496) class, so this call will NOT move the joint dynamically for a static **Model** (p. 624).

But if this **Model** (p. 624) is not static, then it is updated dynamically, all the conencted children **Link** (p. 542)'s are moved as a result of the **Joint** (p. 496) angle setting. Dynamic **Joint** (p. 496) angle update is accomplished by calling **JointController::SetJointPosition** (p. 522).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Angle to set the joint to. |

#### 10.86.4.42 virtual void gazebo::physics::Joint::SetAttribute ( const std::string & *_key*, int *_index*, const boost::any & *_value* ) [pure virtual]

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

**Parameters**

| | | |
|---|---|---|
| in | *_key* | String key. |
| in | *_index* | Index of the axis. |
| in | *_value* | Value of the attribute. |

Implemented in **gazebo::physics::DARTJoint** (p. 312), and **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.86.4.43 virtual void gazebo::physics::Joint::SetAxis ( int *_index*, const math::Vector3 & *_axis* ) [pure virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Implemented in **gazebo::physics::SimbodyJoint** (p. 898), **gazebo::physics::BallJoint< DARTJoint >** (p. 159), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 159), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::DARTHinge2Joint** (p. 298), **gazebo::physics::DARTHingeJoint** (p. 303), **gazebo::physics::SimbodyScrewJoint** (p. 934), **gazebo::physics::SimbodyHingeJoint** (p. 890), **gazebo::physics::DARTSliderJoint** (p. 349), **gazebo::physics::DARTUniversalJoint** (p. 356), **gazebo::physics::DARTScrewJoint** (p. 343), and **gazebo::physics::SimbodySliderJoint** (p. 941).

#### 10.86.4.44 virtual void gazebo::physics::Joint::SetDamping ( int *_index*, double *_damping* ) [pure virtual]

Set the joint damping.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set, currently ignored, to be implemented. |
| in | *_damping* | Damping value for the axis. |

Implemented in **gazebo::physics::DARTJoint** (p. 312), **gazebo::physics::SimbodyJoint** (p. 898), **gazebo::physics::SimbodySliderJoint** (p. 941), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyHingeJoint** (p. 890), **gazebo::physics::SimbodyScrewJoint** (p. 934), and **gazebo::physics::SimbodyBallJoint** (p. 869).

#### 10.86.4.45 virtual void gazebo::physics::Joint::SetEffortLimit ( unsigned int *_index*, double *_effort* ) [virtual]

Set the effort limit on a joint axis.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_effort* | Effort limit for the axis. |

#### 10.86.4.46 virtual void gazebo::physics::Joint::SetForce ( int *_index*, double *_effort* ) [pure virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)). Forces are truncated by effortLimit before applied.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_effort* | Force value. |

Implemented in **gazebo::physics::DARTJoint** (p. 312), and **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.86.4.47 virtual void gazebo::physics::Joint::SetHighStop ( int *_index*, const math::Angle & *_angle* ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented in **gazebo::physics::SimbodyHinge2Joint** (p. 885), **gazebo::physics::SimbodyUniversalJoint** (p. 950), **gazebo::physics::DARTJoint** (p. 313), **gazebo::physics::SimbodyBallJoint** (p. 870), **gazebo::physics::SimbodyHingeJoint** (p. 891), **gazebo::physics::SimbodySliderJoint** (p. 941), and **gazebo::physics::SimbodyScrewJoint** (p. 935).

#### 10.86.4.48 virtual void gazebo::physics::Joint::SetLowStop ( int *_index*, const math::Angle & *_angle* ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented in **gazebo::physics::SimbodyHinge2Joint** (p. 885), **gazebo::physics::SimbodyUniversalJoint** (p. 950), **gazebo::physics::DARTJoint** (p. 313), **gazebo::physics::SimbodyBallJoint** (p. 870), **gazebo::physics::SimbodyHingeJoint** (p. 891), **gazebo::physics::SimbodySliderJoint** (p. 942), and **gazebo::physics::SimbodyScrewJoint** (p. 935).

#### 10.86.4.49 virtual void gazebo::physics::Joint::SetMaxForce ( int *_index*, double *_force* ) [pure virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 935), **gazebo::physics::SimbodyHinge2Joint** (p. 885), **gazebo::physics::SimbodyUniversalJoint** (p. 950), **gazebo::physics::SimbodyBallJoint** (p. 870), **gazebo::physics::DARTHinge2Joint** (p. 299), **gazebo::physics::DARTHingeJoint** (p. 304), **gazebo::physics::DARTScrewJoint** (p. 344), **gazebo::physics::DARTSliderJoint** (p. 349), **gazebo::physics::DARTUniversalJoint** (p. 357), **gazebo::physics::SimbodyHingeJoint** (p. 891), **gazebo::physics::DARTBallJoint** (p. 284), and **gazebo::physics::SimbodySliderJoint** (p. 942).

#### 10.86.4.50 void gazebo::physics::Joint::SetModel ( ModelPtr _model )

Set the model this joint belongs too.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to a model. |

#### 10.86.4.51 virtual void gazebo::physics::Joint::SetProvideFeedback ( bool _enable ) [virtual]

Set whether the joint should generate feedback.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | True to enable joint feedback. |

#### 10.86.4.52 void gazebo::physics::Joint::SetState ( const JointState & _state )

Set the joint state.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **Joint** (p. 496) state |

#### 10.86.4.53 virtual void gazebo::physics::Joint::SetVelocity ( int _index, double _vel ) [pure virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 936), **gazebo::physics::SimbodyHinge2Joint** (p. 885), **gazebo::physics::SimbodyUniversalJoint** (p. 951), **gazebo::physics::DARTHinge2Joint** (p. 299), **gazebo::physics::DARTScrewJoint** (p. 344), **gazebo::physics::DARTHingeJoint** (p. 304), **gazebo::physics::DARTUniversalJoint** (p. 357), **gazebo::physics::SimbodyBallJoint** (p. 871), **gazebo::physics::DARTSliderJoint** (p. 349), **gazebo::physics::SimbodyHingeJoint** (p. 891), **gazebo::physics::DARTBallJoint** (p. 284), and **gazebo::physics::SimbodySliderJoint** (p. 942).

#### 10.86.4.54 void gazebo::physics::Joint::Update ( ) [virtual]

Update the joint.

Reimplemented from **gazebo::physics::Base** (p. 171).

#### 10.86.4.55 virtual void gazebo::physics::Joint::UpdateParameters ( sdf::ElementPtr _sdf ) [virtual]

Update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to update from. |

Reimplemented from **gazebo::physics::Base** (p. 171).

### 10.86.5 Member Data Documentation

#### 10.86.5.1 LinkPtr gazebo::physics::Joint::anchorLink [protected]

Anchor link.

#### 10.86.5.2 math::Vector3 gazebo::physics::Joint::anchorPos [protected]

Anchor pose.

This is the xyz offset of the joint frame from child frame specified in the parent link frame

#### 10.86.5.3 math::Pose gazebo::physics::Joint::anchorPose [protected]

Anchor pose specified in SDF <joint><pose> tag.

AnchorPose is the transform from child link frame to joint frame specified in the child link frame. AnchorPos is more relevant in normal usage, but sometimes, we do need this (e.g. GetForceTorque and joint visualization).

#### 10.86.5.4 gazebo::event::ConnectionPtr gazebo::physics::Joint::applyDamping [protected]

apply damping for adding viscous damping forces on updates

#### 10.86.5.5 LinkPtr gazebo::physics::Joint::childLink [protected]

The first link this joint connects to.

#### 10.86.5.6 double gazebo::physics::Joint::dampingCoefficient [protected]

joint dampingCoefficient

#### 10.86.5.7 double gazebo::physics::Joint::effortLimit[2] [protected]

Store **Joint** (p. 496) effort limit as specified in SDF.

#### 10.86.5.8 double gazebo::physics::Joint::inertiaRatio[2] [protected]

Store **Joint** (p. 496) inertia ratio.

This is a measure of how well this model behaves using interative LCP solvers.

#### 10.86.5.9 math::Angle gazebo::physics::Joint::lowerLimit[2] [protected]

Store **Joint** (p. 496) position lower limit as specified in SDF.

#### 10.86.5.10 ModelPtr gazebo::physics::Joint::model [protected]

Pointer to the parent model.

#### 10.86.5.11 LinkPtr gazebo::physics::Joint::parentLink [protected]

The second link this joint connects to.

#### 10.86.5.12 bool gazebo::physics::Joint::provideFeedback [protected]

Provide Feedback data for contact forces.

#### 10.86.5.13 math::Angle gazebo::physics::Joint::upperLimit[2] [protected]

Store **Joint** (p. 496) position upper limit as specified in SDF.

#### 10.86.5.14 bool gazebo::physics::Joint::useCFMDamping [protected]

option to use CFM damping

#### 10.86.5.15 double gazebo::physics::Joint::velocityLimit[2] [protected]

Store **Joint** (p. 496) velocity limit as specified in SDF.

#### 10.86.5.16 JointWrench gazebo::physics::Joint::wrench [protected]

Cache **Joint** (p. 496) force torque values in case physics engine clears them at the end of update step.

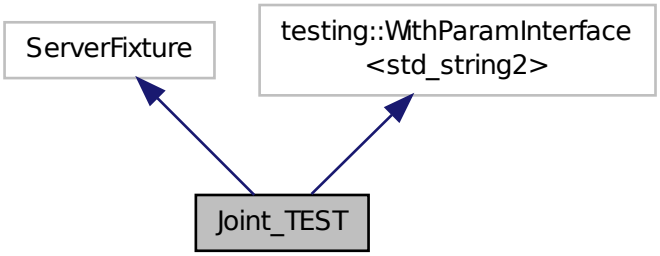
The documentation for this class was generated from the following file:

- **Joint.hh**

## 10.87 Joint_TEST Class Reference

`#include <Joint_TEST.hh>`

Inheritance diagram for Joint_TEST:




## Classes

- class **SpawnJointOptions**

  *Class to hold parameters for spawning joints.*

## Public Member Functions

- void **ForceTorque1** (const std::string &_physicsEngine)

  *Load example world with a few joints Measure / verify static force torques against analytical answers.*
- void **ForceTorque2** (const std::string &_physicsEngine)

  *Load example world with a few joints Measure / verify static force torques against analytical answers.*
- void **GetForceTorqueWithAppliedForce** (const std::string &_physicsEngine)

  *Load example world with a few joints.*
- void **JointCreationDestructionTest** (const std::string &_physicsEngine)

  *Create and destroy joints repeatedly, monitors memory usage.*
- void **JointTorqueTest** (const std::string &_physicsEngine)

  *Create a hinge joint between link and world.*
- virtual void **SetUp** ()
- **physics::JointPtr SpawnJoint** (const std::string &_type, bool _worldChild=false, bool _worldParent=false, **common::Time** _wait=**common::Time**(99, 0))

  *Spawn a model with a joint connecting to the world.*
- **physics::JointPtr SpawnJoint** (const **SpawnJointOptions** &_opt)

  *Spawn a model with a joint connecting to the world.*
- void **SpawnJointRotational** (const std::string &_physicsEngine, const std::string &_jointType)

  *Spawn model with rotational joints.*
- void **SpawnJointRotationalWorld** (const std::string &_physicsEngine, const std::string &_jointType)

  *Spawn model with rotational joints.*
- void **SpawnJointTypes** (const std::string &_physicsEngine, const std::string &_jointType)

  *Spawn model with each type of joint.*

### Protected Member Functions

- **Joint_TEST** ()

### Protected Attributes

- std::string **jointType**

  *Joint type for test.*
- std::string **physicsEngine**

  *Physics engine for test.*

## 10.87.1 Constructor & Destructor Documentation

#### 10.87.1.1 Joint_TEST::Joint_TEST ( ) [inline],[protected]

## 10.87.2 Member Function Documentation

#### 10.87.2.1 void Joint_TEST::ForceTorque1 ( const std::string & *_physicsEngine* )

Load example world with a few joints Measure / verify static force torques against analytical answers.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |

#### 10.87.2.2 void Joint_TEST::ForceTorque2 ( const std::string & *_physicsEngine* )

Load example world with a few joints Measure / verify static force torques against analytical answers.

Change gravity to tip over the joints. Wait until joint stops are hit and joint motion settles, then check force torques values against analytical values.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |

#### 10.87.2.3 void Joint_TEST::GetForceTorqueWithAppliedForce ( const std::string & *_physicsEngine* )

Load example world with a few joints.

Servo the joints to a fixed target position using simple PID controller. Measure / verify static force torques against analytical answers.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |

#### 10.87.2.4 void Joint_TEST::JointCreationDestructionTest ( const std::string & _physicsEngine )

Create and destroy joints repeatedly, monitors memory usage.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |

#### 10.87.2.5 void Joint_TEST::JointTorqueTest ( const std::string & _physicsEngine )

Create a hinge joint between link and world.

Apply force and check acceleration against analytical solution.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |

#### 10.87.2.6 virtual void Joint_TEST::SetUp ( ) [inline], [virtual]

References gzdbg.

#### 10.87.2.7 physics::JointPtr Joint_TEST::SpawnJoint ( const std::string & _type, bool _worldChild = false, bool _worldParent = false, common::Time _wait = common::Time (99, 0) ) [inline]

Spawn a model with a joint connecting to the world.

The function will wait for duration _wait for the model to spawn and attempt to return a pointer to the spawned joint. This function is not guaranteed to return a valid JointPtr, so the output should be checked.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | Type of joint to create. |
| in | *_worldChild* | Flag to set child link to the world. |
| in | *_worldParent* | Flag to set parent link to the world. |
| in | *_wait* | Length of time to wait for model to spawn in order to return Joint pointer. |

References Joint_TEST::SpawnJointOptions::type, Joint_TEST::SpawnJointOptions::wait, Joint_TEST::SpawnJointOptions::worldChild, and Joint_TEST::SpawnJointOptions::worldParent.

#### 10.87.2.8 physics::JointPtr Joint_TEST::SpawnJoint ( const SpawnJointOptions & _opt ) [inline]

Spawn a model with a joint connecting to the world.

The function will wait for duration _wait for the model to spawn and attempt to return a pointer to the spawned joint. This function is not guaranteed to return a valid JointPtr, so the output should be checked.

**Parameters**

| | | |
|---|---|---|
| in | *_opt* | Options for spawned model and joint. |

References Joint_TEST::SpawnJointOptions::axis, Joint_TEST::SpawnJointOptions::childLinkPose, gazebo::physics-

::get_world(), gazebo::common::Time::GetWallTime(), gzwarn, Joint_TEST::SpawnJointOptions::jointPose, Joint_TEST::SpawnJointOptions::modelPose, gazebo::common::Time::MSleep(), Joint_TEST::SpawnJointOptions::noLinkPose, NULL, Joint_TEST::SpawnJointOptions::parentLinkPose, Joint_TEST::SpawnJointOptions::type, Joint_TEST::SpawnJointOptions::wait, Joint_TEST::SpawnJointOptions::worldChild, Joint_TEST::SpawnJointOptions::worldParent, and gazebo::common::Time::Zero.

#### 10.87.2.9 void Joint_TEST::SpawnJointRotational ( const std::string & *_physicsEngine,* const std::string & *_jointType* )

Spawn model with rotational joints.

Set velocity on parent and make sure child follows.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |
| in | *_jointType* | Type of joint to spawn and test. |

#### 10.87.2.10 void Joint_TEST::SpawnJointRotationalWorld ( const std::string & *_physicsEngine,* const std::string & *_jointType* )

Spawn model with rotational joints.

Attach to world and make sure it doesn't fall.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |
| in | *_jointType* | Type of joint to spawn and test. |

#### 10.87.2.11 void Joint_TEST::SpawnJointTypes ( const std::string & *_physicsEngine,* const std::string & *_jointType* )

Spawn model with each type of joint.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Type of physics engine to use. |
| in | *_jointType* | Type of joint to spawn and test. |

### 10.87.3 Member Data Documentation

#### 10.87.3.1 std::string Joint_TEST::jointType [protected]

Joint type for test.

#### 10.87.3.2 std::string Joint_TEST::physicsEngine [protected]

Physics engine for test.

The documentation for this class was generated from the following file:

- **Joint_TEST.hh**

# 10.88 gazebo::physics::JointController Class Reference

A class for manipulating **physics::Joint** (p. 496).

`#include <physics/physics.hh>`

## Public Member Functions

- **JointController** (**ModelPtr** _model)

  *Constructor.*
- void **AddJoint** (**JointPtr** _joint)

  *Add a joint to control.*
- void **Reset** ()

  *Reset all commands.*
- void **SetJointPosition** (const std::string &_name, double _position, int _index=0)

  *Set the positions of a **Joint** (p. 496) by name.*
- void **SetJointPosition** (**JointPtr** _joint, double _position, int _index=0)

  *Set the positions of a **Joint** (p. 496) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 973) and radians for **HingeJoint** (p. 472), etc.*
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

  *Set the positions of a set of **Joint** (p. 496)'s.*
- void **Update** ()

  *Update the joint control.*

### 10.88.1 Detailed Description

A class for manipulating **physics::Joint** (p. 496).

### 10.88.2 Constructor & Destructor Documentation

#### 10.88.2.1 gazebo::physics::JointController::JointController ( **ModelPtr** *_model* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | **Model** (p. 624) that uses this joint controller. |

### 10.88.3 Member Function Documentation

#### 10.88.3.1 void gazebo::physics::JointController::AddJoint ( **JointPtr** *_joint* )

Add a joint to control.

**Parameters**

| | | |
|---|---|---|
| in | *_joint* | **Joint** (p. 496) to control. |

#### 10.88.3.2 void gazebo::physics::JointController::Reset ( )

Reset all commands.

#### 10.88.3.3 void gazebo::physics::JointController::SetJointPosition ( const std::string & *_name,* double *_position,* int *_index* = 0 )

Set the positions of a **Joint** (p. 496) by name.

**See Also**

JointController::SetJointPosition(JointPtr, double)

#### 10.88.3.4 void gazebo::physics::JointController::SetJointPosition ( JointPtr *_joint,* double *_position,* int *_index* = 0 )

Set the positions of a **Joint** (p. 496) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 973) and radians for **HingeJoint** (p. 472), etc.

Implementation: In order to change the position of a **Joint** (p. 496) inside a **Model** (p. 624), this call must recursively crawl through all the connected children **Link** (p. 542)'s in this **Model** (p. 624), and update each **Link** (p. 542) Pose affected by this **Joint** (p. 496) angle update. Warning: There is no constraint satisfaction being done here, traversal through the kinematic graph has unexpected behavior if you try to set the joint position of a link inside a loop structure.

**Parameters**

| | | |
|---|---|---|
| in | *_joint* | **Joint** (p. 496) to set. |
| in | *_position* | Position of the joint. |

#### 10.88.3.5 void gazebo::physics::JointController::SetJointPositions ( const std::map< std::string, double > & *_jointPositions* )

Set the positions of a set of **Joint** (p. 496)'s.

**See Also**

JointController::SetJointPosition(JointPtr, double)

#### 10.88.3.6 void gazebo::physics::JointController::Update ( )

Update the joint control.

The documentation for this class was generated from the following file:

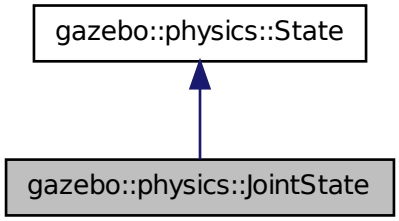
- **JointController.hh**

## 10.89 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 496)

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::JointState:




## Public Member Functions

- **JointState** ()
  *Default constructor.*
- **JointState** (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)
  *Constructor.*
- **JointState** (**JointPtr** _joint)
  *Constructor.*
- **JointState** (const sdf::ElementPtr _sdf)
  *Constructor.*
- virtual ∼**JointState** ()
  *Destructor.*
- void **FillSDF** (sdf::ElementPtr _sdf)
  *Populate a state SDF element with data from the object.*
- **math::Angle GetAngle** (unsigned int _axis) const
  *Get the joint angle.*
- unsigned int **GetAngleCount** () const
  *Get the number of angles.*
- const std::vector< **math::Angle** > & **GetAngles** () const
  *Get the angles.*
- bool **IsZero** () const
  *Return true if the values in the state are zero.*
- void **Load** (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)
  *Load.*
- virtual void **Load** (const sdf::ElementPtr _elem)
  *Load state from SDF element.*
- **JointState operator+** (const **JointState** &_state) const
  *Addition operator.*
- **JointState operator-** (const **JointState** &_state) const
  *Subtraction operator.*
- **JointState** & **operator=** (const **JointState** &_state)
  *Assignment operator.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::JointState** &_state)
  *Stream insertion operator.*

## Additional Inherited Members

### 10.89.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 496)

### 10.89.2 Constructor & Destructor Documentation

#### 10.89.2.1 gazebo::physics::JointState::JointState ( )

Default constructor.

#### 10.89.2.2 gazebo::physics::JointState::JointState ( JointPtr *_joint,* const common::Time & *_realTime,* const common::Time & *_simTime* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_joint* | **Joint** (p. 496) to get the state of. |
| in | *_realTime* | Real time stamp. |
| in | *_simTime* | Sim time stamp. |

#### 10.89.2.3 gazebo::physics::JointState::JointState ( JointPtr *_joint* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_joint* | **Joint** (p. 496) to get the state of. |

#### 10.89.2.4 gazebo::physics::JointState::JointState ( const sdf::ElementPtr *_sdf* ) [explicit]

Constructor.

Build a **JointState** (p. 522) from SDF data

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF data to load a joint state from. |

#### 10.89.2.5 virtual gazebo::physics::JointState::~JointState ( ) [virtual]

Destructor.

### 10.89.3 Member Function Documentation

#### 10.89.3.1 void gazebo::physics::JointState::FillSDF ( sdf::ElementPtr _sdf )

Populate a state SDF element with data from the object.

**Parameters**

| | | |
|---|---|---|
| out | *_sdf* | SDF element to populate. |

#### 10.89.3.2 math::Angle gazebo::physics::JointState::GetAngle ( unsigned int _axis ) const

Get the joint angle.

**Parameters**

| | | |
|---|---|---|
| in | *_axis* | The axis index. |

**Returns**

Angle of the axis.

**Exceptions**

| | |
|---|---|
| ***common::Exception*** *(p. 416)* | When _axis is invalid. |

#### 10.89.3.3 unsigned int gazebo::physics::JointState::GetAngleCount ( ) const

Get the number of angles.

**Returns**

The number of angles.

#### 10.89.3.4 const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles ( ) const

Get the angles.

**Returns**

Vector of angles.

#### 10.89.3.5 bool gazebo::physics::JointState::IsZero ( ) const

Return true if the values in the state are zero.

**Returns**

True if the values in the state are zero.

#### 10.89.3.6 void gazebo::physics::JointState::Load ( JointPtr *_joint,* const common::Time & *_realTime,* const common::Time & *_simTime* )

Load.

**Parameters**

| | | |
|---|---|---|
| in | *_joint* | **Joint** (p. 496) to get the state of. |
| in | *_realTime* | Real time stamp. |
| in | *_simTime* | Sim time stamp. |

#### 10.89.3.7 virtual void gazebo::physics::JointState::Load ( const sdf::ElementPtr *_elem* ) [virtual]

Load state from SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | SDf values to load from. |

Reimplemented from **gazebo::physics::State** (p. 1000).

#### 10.89.3.8 JointState gazebo::physics::JointState::operator+ ( const JointState & *_state* ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to add. |

**Returns**

The resulting state.

#### 10.89.3.9 JointState gazebo::physics::JointState::operator- ( const JointState & *_state* ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to substract. |

**Returns**

The resulting state.

#### 10.89.3.10 JointState& gazebo::physics::JointState::operator= ( const JointState & _state )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **State** (p. 998) value |

**Returns**

this

### 10.89.4 Friends And Related Function Documentation

#### 10.89.4.1 std::ostream& operator<< ( std::ostream & _out, const gazebo::physics::JointState & _state ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream. |
| in | *_state* | **Joint** (p. 496) state to output. |

**Returns**

The stream.

The documentation for this class was generated from the following file:

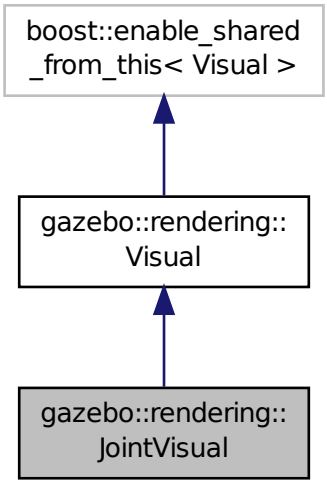
- **JointState.hh**

## 10.90 gazebo::rendering::JointVisual Class Reference

Visualization for joints.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::JointVisual:




### Public Member Functions

- **JointVisual** (const std::string &_name, **VisualPtr** _vis)

  *Constructor.*
- virtual ~**JointVisual** ()

  *Destructor.*
- void **Load** (ConstJointPtr &_msg)

  *Load the visual based on a message.*

### Additional Inherited Members

## 10.90.1 Detailed Description

Visualization for joints.

## 10.90.2 Constructor & Destructor Documentation

### 10.90.2.1 gazebo::rendering::JointVisual::JointVisual ( const std::string & *_name*, VisualPtr *_vis* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual |
| in | *_vis* | Pointer to the parent visual |

#### 10.90.2.2 virtual gazebo::rendering::JointVisual::~JointVisual ( ) [virtual]

Destructor.

### 10.90.3 Member Function Documentation

#### 10.90.3.1 void gazebo::rendering::JointVisual::Load ( ConstJointPtr & _msg )

Load the visual based on a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Joint message |

The documentation for this class was generated from the following file:

- **JointVisual.hh**

## 10.91 gazebo::physics::JointWrench Class Reference

Wrench information from a joint.

`#include <physics/physics.hh>`

### Public Member Functions

- **JointWrench** & **operator+** (const **JointWrench** &_wrench)
  *Operator +.*
- **JointWrench** & **operator-** (const **JointWrench** &_wrench)
  *Operator -.*
- **JointWrench** & **operator=** (const **JointWrench** &_wrench)
  *Operator =.*

### Public Attributes

- **math::Vector3 body1Force**
  *Force on the first link.*
- **math::Vector3 body1Torque**
  *Torque on the first link.*
- **math::Vector3 body2Force**
  *Force on the second link.*
- **math::Vector3 body2Torque**
  *Torque on the second link.*

### 10.91.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Joint** (p. 496) frame immediately after rotation.

## 10.91.2 Member Function Documentation

### 10.91.2.1 JointWrench& gazebo::physics::JointWrench::operator+ ( const JointWrench & _wrench ) [inline]

Operator +.

**Parameters**

| | | |
|---|---|---|
| in | *_wrench* | **Joint** (p. 496) wrench to add |

**Returns**

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

### 10.91.2.2 JointWrench& gazebo::physics::JointWrench::operator- ( const JointWrench & _wrench ) [inline]

Operator -.

**Parameters**

| | | |
|---|---|---|
| in | *_wrench* | **Joint** (p. 496) wrench to subtract |

**Returns**

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

### 10.91.2.3 JointWrench& gazebo::physics::JointWrench::operator= ( const JointWrench & _wrench ) [inline]

Operator =.

**Parameters**

| | | |
|---|---|---|
| in | *_wrench* | **Joint** (p. 496) wrench to set from. |

**Returns**

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

## 10.91.3 Member Data Documentation

### 10.91.3.1 math::Vector3 gazebo::physics::JointWrench::body1Force

Force on the first link.

Referenced by operator+(), operator-(), and operator=().

#### 10.91.3.2 math::Vector3 gazebo::physics::JointWrench::body1Torque

Torque on the first link.

Referenced by operator+(), operator-(), and operator=().

#### 10.91.3.3 math::Vector3 gazebo::physics::JointWrench::body2Force

Force on the second link.

Referenced by operator+(), operator-(), and operator=().

#### 10.91.3.4 math::Vector3 gazebo::physics::JointWrench::body2Torque

Torque on the second link.

Referenced by operator+(), operator-(), and operator=().

The documentation for this class was generated from the following file:

- **JointWrench.hh**

## 10.92 gazebo::common::KeyEvent Class Reference

Generic description of a keyboard event.

`#include <common/common.hh>`

### Public Types

- enum **EventType** { **NO_EVENT**, **PRESS**, **RELEASE** }

  *Key event types enumeration.*

### Public Member Functions

- **KeyEvent** ()

  *Constructor.*

### Public Attributes

- int **key**
- **EventType type**

  *Event type.*

### 10.92.1 Detailed Description

Generic description of a keyboard event.

### 10.92.2 Member Enumeration Documentation

#### 10.92.2.1 enum gazebo::common::KeyEvent::EventType

Key event types enumeration.

**Enumerator**

***NO_EVENT***

***PRESS***

***RELEASE***

### 10.92.3 Constructor & Destructor Documentation

#### 10.92.3.1 gazebo::common::KeyEvent::KeyEvent ( ) [inline]

Constructor.

### 10.92.4 Member Data Documentation

#### 10.92.4.1 int gazebo::common::KeyEvent::key

#### 10.92.4.2 EventType gazebo::common::KeyEvent::type

Event type.

The documentation for this class was generated from the following file:

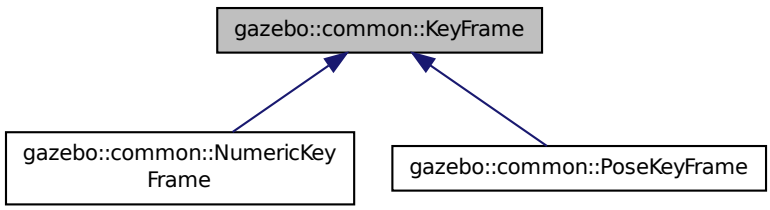
- **KeyEvent.hh**

## 10.93 gazebo::common::KeyFrame Class Reference

A key frame in an animation.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::KeyFrame:



### Public Member Functions

- **KeyFrame** (double _time)
  *Constructor.*
- virtual ~**KeyFrame** ()
  *Destructor.*
- double **GetTime** () const
  *Get the time of the keyframe.*

### Protected Attributes

- double **time**
  *time of key frame*

## 10.93.1 Detailed Description

A key frame in an animation.

## 10.93.2 Constructor & Destructor Documentation

### 10.93.2.1 gazebo::common::KeyFrame::KeyFrame ( double *_time* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | **Time** (p. 1031) of the keyframe in seconds |

### 10.93.2.2 virtual gazebo::common::~KeyFrame ( ) [virtual]

Destructor.

## 10.93.3 Member Function Documentation

### 10.93.3.1 double gazebo::common::KeyFrame::GetTime ( ) const

Get the time of the keyframe.

**Returns**

the time

## 10.93.4 Member Data Documentation

### 10.93.4.1 double gazebo::common::KeyFrame::time [protected]

time of key frame

The documentation for this class was generated from the following file:

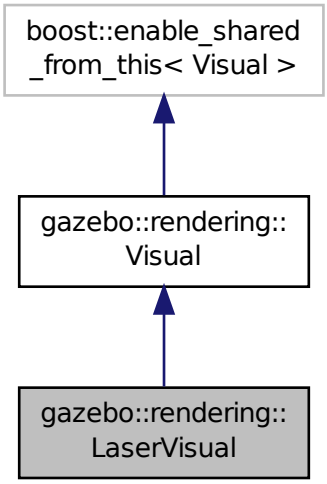
- **KeyFrame.hh**

## 10.94 gazebo::rendering::LaserVisual Class Reference

Visualization for laser data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::LaserVisual:




### Public Member Functions

- **LaserVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  *Constructor.*
- virtual ~**LaserVisual** ()

  *Destructor.*
- virtual void **SetEmissive** (const **common::Color** &_color)

  *Documentation inherited from parent.*

### Additional Inherited Members

### 10.94.1 Detailed Description

Visualization for laser data.

### 10.94.2 Constructor & Destructor Documentation

#### 10.94.2.1 gazebo::rendering::LaserVisual::LaserVisual ( const std::string & *_name*, VisualPtr *_vis*, const std::string & *_topicName* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual. |
| in | *_vis* | Pointer to the parent **Visual** (p. 1121). |
| in | *_topicName* | Name of the topic that has laser data. |

#### 10.94.2.2 virtual gazebo::rendering::LaserVisual::~LaserVisual ( ) [virtual]

Destructor.

### 10.94.3 Member Function Documentation

#### 10.94.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive ( const common::Color & *_color* ) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 1137).

The documentation for this class was generated from the following file:

- **LaserVisual.hh**

## 10.95 gazebo::rendering::Light Class Reference

A light source.

`#include <rendering/rendering.hh>`

### Public Member Functions

- **Light** (**ScenePtr** _scene)
  *Constructor.*
- virtual ~**Light** ()
  *Destructor.*
- void **FillMsg** (msgs::Light &_msg) const
  *Fill the contents of a light message.*
- **common::Color GetDiffuseColor** () const
  *Get the diffuse color.*
- **math::Vector3 GetDirection** () const
  *Get the direction.*
- std::string **GetName** () const
  *Get the name of the visual.*

- **math::Vector3 GetPosition** () const

  *Get the position of the light.*
- **common::Color GetSpecularColor** () const

  *Get the specular color.*
- std::string **GetType** () const

  *Get the type of the light.*
- bool **GetVisible** () const

  *Get whether the light is visible.*
- void **Load** (sdf::ElementPtr _sdf)

  *Load the light using a set of SDF parameters.*
- void **Load** ()

  *Load the light using default parameters.*
- void **LoadFromMsg** (ConstLightPtr &_msg)

  *Load from a light message.*
- void **SetAttenuation** (double _constant, double _linear, double _quadratic)

  *Set the attenuation.*
- void **SetCastShadows** (const bool &_cast)

  *Set cast shadows.*
- void **SetDiffuseColor** (const **common::Color** &_color)

  *Set the diffuse color.*
- void **SetDirection** (const **math::Vector3** &_dir)

  *Set the direction.*
- void **SetLightType** (const std::string &_type)

  *Set the light type.*
- void **SetName** (const std::string &_name)

  *Set the name of the visual.*
- void **SetPosition** (const **math::Vector3** &_p)

  *Set the position of the light.*
- void **SetRange** (const double &_range)

  *Set the range.*
- virtual bool **SetSelected** (bool _s)

  *Set whether this entity has been selected by the user through the gui.*
- void **SetSpecularColor** (const **common::Color** &_color)

  *Set the specular color.*
- void **SetSpotFalloff** (const double &_value)

  *Set the spot light falloff.*
- void **SetSpotInnerAngle** (const double &_angle)

  *Set the spot light inner angle.*
- void **SetSpotOuterAngle** (const double &_angle)

  *Set the spot light outer angle.*
- void **ShowVisual** (bool _s)

  *Set whether to show the visual.*
- void **ToggleShowVisual** ()
- void **UpdateFromMsg** (ConstLightPtr &_msg)

  *Update a light source from a message.*

### Protected Member Functions

- virtual void **OnPoseChange** ()

  *On pose change callback.*

## 10.95.1 Detailed Description

A light source.

There are three types of lights: Point, Spot, and Directional. This class encapsulates all three. Point lights are light light bulbs, spot lights project a cone of light, and directional lights are light sun light.

## 10.95.2 Constructor & Destructor Documentation

#### 10.95.2.1 gazebo::rendering::Light::Light ( ScenePtr *_scene* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | Pointer to the scene that contains the **Light** (p. 535). |

#### 10.95.2.2 virtual gazebo::rendering::Light::~Light ( ) [virtual]

Destructor.

## 10.95.3 Member Function Documentation

#### 10.95.3.1 void gazebo::rendering::Light::FillMsg ( msgs::Light & *_msg* ) const

Fill the contents of a light message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message to fill. |

#### 10.95.3.2 common::Color gazebo::rendering::Light::GetDiffuseColor ( ) const

Get the diffuse color.

**Returns**

The light's diffuse color.

#### 10.95.3.3 math::Vector3 gazebo::rendering::Light::GetDirection ( ) const

Get the direction.

**Returns**

The light's direction.

#### 10.95.3.4 std::string gazebo::rendering::Light::GetName ( ) const

Get the name of the visual.

**Returns**

The light's name.

#### 10.95.3.5 math::Vector3 gazebo::rendering::Light::GetPosition ( ) const

Get the position of the light.

**Returns**

The position of the light

#### 10.95.3.6 common::Color gazebo::rendering::Light::GetSpecularColor ( ) const

Get the specular color.

**Returns**

The specular color

#### 10.95.3.7 std::string gazebo::rendering::Light::GetType ( ) const

Get the type of the light.

**Returns**

The light type: "point", "spot", "directional".

#### 10.95.3.8 bool gazebo::rendering::Light::GetVisible ( ) const

Get whether the light is visible.

**Returns**

True if the light is visible.

#### 10.95.3.9 void gazebo::rendering::Light::Load ( sdf::ElementPtr *_sdf* )

Load the light using a set of SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to the SDF containing the **Light** (p. 535) description. |

#### 10.95.3.10 void gazebo::rendering::Light::Load ( )

Load the light using default parameters.

#### 10.95.3.11 void gazebo::rendering::Light::LoadFromMsg ( ConstLightPtr & *_msg* )

Load from a light message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Containing the light information. |

#### 10.95.3.12 virtual void gazebo::rendering::Light::OnPoseChange ( ) [inline], [protected], [virtual]

On pose change callback.

#### 10.95.3.13 void gazebo::rendering::Light::SetAttenuation ( double *_constant,* double *_linear,* double *_quadratic* )

Set the attenuation.

**Parameters**

| | | |
|---|---|---|
| in | *_contant* | Constant attenuation |
| in | *_linear* | Linear attenuation |
| in | *_quadratic* | Quadratic attenuation |

#### 10.95.3.14 void gazebo::rendering::Light::SetCastShadows ( const bool & *_cast* )

Set cast shadows.

**Parameters**

| | | |
|---|---|---|
| in | *_cast* | Set to true to cast shadows. |

#### 10.95.3.15 void gazebo::rendering::Light::SetDiffuseColor ( const common::Color & *_color* )

Set the diffuse color.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | **Light** (p. 535) diffuse color. |

#### 10.95.3.16 void gazebo::rendering::Light::SetDirection ( const math::Vector3 & _dir )

Set the direction.

**Parameters**

| | | |
|---|---|---|
| in | *_dir* | Set the light's direction. Only applicable to spot and directional lights. |

#### 10.95.3.17 void gazebo::rendering::Light::SetLightType ( const std::string & _type )

Set the light type.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | The light type: "point", "spot", "directional" |

#### 10.95.3.18 void gazebo::rendering::Light::SetName ( const std::string & _name )

Set the name of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the light source. |

#### 10.95.3.19 void gazebo::rendering::Light::SetPosition ( const math::Vector3 & _p )

Set the position of the light.

**Parameters**

| | | |
|---|---|---|
| in | *_p* | New position for the light |

#### 10.95.3.20 void gazebo::rendering::Light::SetRange ( const double & _range )

Set the range.

**Parameters**

| | | |
|---|---|---|
| in | *_range* | Rage of the light in meters. |

#### 10.95.3.21 virtual bool gazebo::rendering::Light::SetSelected ( bool _s ) [virtual]

Set whether this entity has been selected by the user through the gui.

**Parameters**

| | | |
|---|---|---|
| in | *_s* | Set to True when the light is selected by the user. |

#### 10.95.3.22 void gazebo::rendering::Light::SetSpecularColor ( const common::Color & _color )

Set the specular color.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | The specular color |

#### 10.95.3.23 void gazebo::rendering::Light::SetSpotFalloff ( const double & _value )

Set the spot light falloff.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | Falloff value |

#### 10.95.3.24 void gazebo::rendering::Light::SetSpotInnerAngle ( const double & _angle )

Set the spot light inner angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | Inner angle in radians |

#### 10.95.3.25 void gazebo::rendering::Light::SetSpotOuterAngle ( const double & _angle )

Set the spot light outer angle.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | Outer angle in radians |

#### 10.95.3.26 void gazebo::rendering::Light::ShowVisual ( bool _s )

Set whether to show the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_s* | Set to true to draw a representation of the light. |

#### 10.95.3.27 void gazebo::rendering::Light::ToggleShowVisual ( )

#### 10.95.3.28 void gazebo::rendering::Light::UpdateFromMsg ( ConstLightPtr & _msg )

Update a light source from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | **Light** (p. 535) message to update from |

The documentation for this class was generated from the following file:

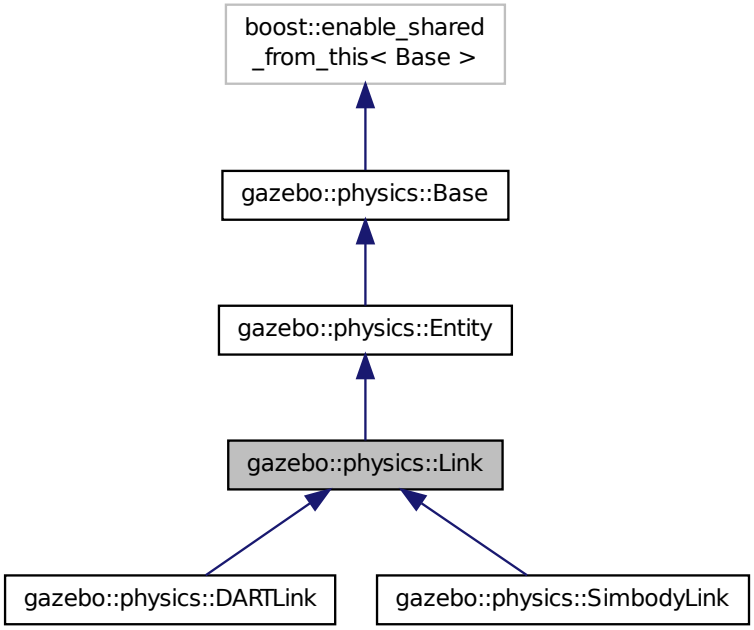
- **Light.hh**

## 10.96 gazebo::physics::Link Class Reference

**Link** (p. 542) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Link:




### Public Member Functions

- **Link** (**EntityPtr** _parent)

  *Constructor.*
- virtual ~**Link** ()

  *Destructor.*
- void **AddChildJoint** (**JointPtr** _joint)

  *Joints that have this **Link** (p. 542) as a parent **Link** (p. 542).*

- virtual void **AddForce** (const **math::Vector3** &_force)=0

  *Add a force to the body.*
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relPos)=0

  *Add a force to the body at position expressed to the body's own frame of reference.*
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)=0

  *Add a force to the body using a global position.*
- void **AddParentJoint** (**JointPtr** _joint)

  *Joints that have this **Link** (p. 542) as a child **Link** (p. 542).*
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)=0

  *Add a force to the body, components are relative to the body's own frame of reference.*
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)=0

  *Add a torque to the body, components are relative to the body's own frame of reference.*
- virtual void **AddTorque** (const **math::Vector3** &_torque)=0

  *Add a torque to the body.*
- void **AttachStaticModel** (**ModelPtr** &_model, const **math::Pose** &_offset)

  *Attach a static model to this link.*
- template<typename T >
  **event::ConnectionPtr ConnectEnabled** (T _subscriber)

  *Connect to the add entity signal.*
- void **DetachAllStaticModels** ()

  *Detach all static models from this link.*
- void **DetachStaticModel** (const std::string &_modelName)

  *Detach a static model from this link.*
- void **DisconnectEnabled** (**event::ConnectionPtr** &_conn)

  *Disconnect to the add entity signal.*
- void **FillMsg** (msgs::Link &_msg)

  *Fill a link message.*
- void **Fini** ()

  *Finalize the body.*
- double **GetAngularDamping** () const

  *Get the angular damping factor.*
- virtual **math::Box GetBoundingBox** () const

  *Get the bounding box for the link and all the child elements.*
- **Joint_V GetChildJoints** () const

  *Get the child joints.*
- **Link_V GetChildJointsLinks** () const

  *Returns a vector of children Links connected by joints.*
- **CollisionPtr GetCollision** (const std::string &_name)

  *Get a child collision by name.*
- **CollisionPtr GetCollision** (unsigned int _index) const

  *Get a child collision by index.*
- **Collision_V GetCollisions** () const

  *Get all the child collisions.*
- virtual bool **GetEnabled** () const =0

  *Get whether this body is enabled in the physics engine.*
- virtual bool **GetGravityMode** () const =0

  *Get the gravity mode.*

- **InertialPtr GetInertial** () const

  *Get the inertia of the link.*
- virtual bool **GetKinematic** () const

  *Implement this function.*
- double **GetLinearDamping** () const

  *Get the linear damping factor.*
- **ModelPtr GetModel** () const

  *Get the model that this body belongs to.*
- **Joint_V GetParentJoints** () const

  *Get the parent joints.*
- **Link_V GetParentJointsLinks** () const

  *Returns a vector of parent Links connected by joints.*
- **math::Vector3 GetRelativeAngularAccel** () const

  *Get the angular acceleration of the body.*
- **math::Vector3 GetRelativeAngularVel** () const

  *Get the angular velocity of the body.*
- **math::Vector3 GetRelativeForce** () const

  *Get the force applied to the body.*
- **math::Vector3 GetRelativeLinearAccel** () const

  *Get the linear acceleration of the body.*
- **math::Vector3 GetRelativeLinearVel** () const

  *Get the linear velocity of the body.*
- **math::Vector3 GetRelativeTorque** () const

  *Get the torque applied to the body.*
- bool **GetSelfCollide** () const

  *Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.*
- unsigned int **GetSensorCount** () const

  *Get sensor count.*
- std::string **GetSensorName** (unsigned int _index) const

  *Get sensor name.*
- **math::Vector3 GetWorldAngularAccel** () const

  *Get the angular acceleration of the body in the world frame.*
- virtual **math::Vector3 GetWorldCoGLinearVel** () const =0

  *Get the linear velocity at the body's center of gravity in the world frame.*
- **math::Pose GetWorldCoGPose** () const

  *Get the pose of the body's center of gravity in the world coordinate frame.*
- virtual **math::Vector3 GetWorldForce** () const =0

  *Get the force applied to the body in the world frame.*
- **math::Vector3 GetWorldLinearAccel** () const

  *Get the linear acceleration of the body in the world frame.*
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset=**math::Vector3**(0, 0, 0)) const =0

  *Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.*
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset, const **math::Quaternion** &_q) const =0

  *Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.*
- virtual **math::Vector3 GetWorldTorque** () const =0

  *Get the torque applied to the body in the world frame.*

- virtual void **Init** ()

  *Initialize the body.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the body based on an SDF element.*
- virtual void **OnPoseChange** ()

  *This function is called when the entity's (or one of its parents) pose of the parent has changed.*
- void **ProcessMsg** (const msgs::Link &_msg)

  *Update parameters from a message.*
- virtual void **RemoveChild** (**EntityPtr** _child)
- void **RemoveChildJoint** (const std::string &_jointName)

  *Remove Joints that have this **Link** (p. 542) as a parent **Link** (p. 542).*
- void **RemoveCollision** (const std::string &_name)

  *Remove a collision from the link.*
- void **RemoveParentJoint** (const std::string &_jointName)

  *Remove Joints that have this **Link** (p. 542) as a child **Link** (p. 542).*
- void **Reset** ()

  *Reset the link.*
- void **ResetPhysicsStates** ()

  *Reset the link.*
- void **SetAngularAccel** (const **math::Vector3** &_accel)

  *Set the angular acceleration of the body.*
- virtual void **SetAngularDamping** (double _damping)=0

  *Set the angular damping factor.*
- virtual void **SetAngularVel** (const **math::Vector3** &_vel)=0

  *Set the angular velocity of the body.*
- virtual void **SetAutoDisable** (bool _disable)=0

  *Allow the link to auto disable.*
- void **SetCollideMode** (const std::string &_mode)

  *Set the collide mode of the body.*
- virtual void **SetEnabled** (bool _enable) const =0

  *Set whether this body is enabled.*
- virtual void **SetForce** (const **math::Vector3** &_force)=0

  *Set the force applied to the body.*
- virtual void **SetGravityMode** (bool _mode)=0

  *Set whether gravity affects this body.*
- void **SetInertial** (const **InertialPtr** &_inertial)

  *Set the mass of the link.*
- virtual void **SetKinematic** (const bool &_kinematic)

  *Implement this function.*
- void **SetLaserRetro** (float _retro)

  *Set the laser retro reflectiveness.*
- void **SetLinearAccel** (const **math::Vector3** &_accel)

  *Set the linear acceleration of the body.*
- virtual void **SetLinearDamping** (double _damping)=0

  *Set the linear damping factor.*
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)=0

  *Set the linear velocity of the body.*

- virtual void **SetLinkStatic** (bool _static)=0

  *Freeze link to ground (inertial frame).*
- void **SetPublishData** (bool _enable)

  *Enable/Disable link data publishing.*
- void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the link.*
- virtual bool **SetSelected** (bool _set)

  *Set whether this entity has been selected by the user through the gui.*
- virtual void **SetSelfCollide** (bool _collide)=0

  *Set whether this body will collide with others in the model.*
- void **SetState** (const **LinkState** &_state)

  *Set the current link state.*
- virtual void **SetTorque** (const **math::Vector3** &_torque)=0

  *Set the torque applied to the body.*
- void **Update** (const **common::UpdateInfo** &_info)

  *Update the collision.*
- virtual void **UpdateMass** ()

  *Update the mass matrix.*
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  *Update the parameters using new sdf values.*
- virtual void **UpdateSurface** ()

  *Update surface parameters.*

## Protected Types

- typedef std::map< uint32_t, msgs::Visual > **Visuals_M**

## Protected Attributes

- **math::Vector3 angularAccel**

  *Angular acceleration.*
- std::vector< **math::Pose** > **attachedModelsOffset**

  *Offsets for the attached models.*
- std::vector< std::string > **cgVisuals**

  *Center of gravity visual elements.*
- **InertialPtr inertial**

  ***Inertial*** *(*p. *483) properties.*
- **math::Vector3 linearAccel**

  *Linear acceleration.*
- **Visuals_M visuals**

  ***Link*** *(*p. *542) visual elements.*

## Additional Inherited Members

### 10.96.1 Detailed Description

**Link** (p. 542) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

### 10.96.2 Member Typedef Documentation

#### 10.96.2.1 typedef std::map<uint32_t, msgs::Visual> gazebo::physics::Link::Visuals_M [protected]

### 10.96.3 Constructor & Destructor Documentation

#### 10.96.3.1 gazebo::physics::Link::Link ( EntityPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of this link. |

#### 10.96.3.2 virtual gazebo::physics::Link::~Link ( ) [virtual]

Destructor.

### 10.96.4 Member Function Documentation

#### 10.96.4.1 void gazebo::physics::Link::AddChildJoint ( JointPtr *_joint* )

Joints that have this **Link** (p. 542) as a parent **Link** (p. 542).

**Parameters**

| | | |
|---|---|---|
| in | *_joint* | **Joint** (p. 496) that is a child of this link. |

#### 10.96.4.2 virtual void gazebo::physics::Link::AddForce ( const math::Vector3 & *_force* ) [pure virtual]

Add a force to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |

Implemented in **gazebo::physics::SimbodyLink** (p. 903), and **gazebo::physics::DARTLink** (p. 317).

#### 10.96.4.3 virtual void gazebo::physics::Link::AddForceAtRelativePosition ( const math::Vector3 & *_force,* const math::Vector3 & *_relPos* ) [pure virtual]

Add a force to the body at position expressed to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |
| in | *_relPos* | Position on the link to add the force. |

Implemented in **gazebo::physics::SimbodyLink** (p. 904), and **gazebo::physics::DARTLink** (p. 317).

#### 10.96.4.4 virtual void gazebo::physics::Link::AddForceAtWorldPosition ( const math::Vector3 & *_force*, const math::Vector3 & *_pos* ) [pure virtual]

Add a force to the body using a global position.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |
| in | *_pos* | Position in global coord frame to add the force. |

Implemented in **gazebo::physics::SimbodyLink** (p. 904), and **gazebo::physics::DARTLink** (p. 317).

#### 10.96.4.5 void gazebo::physics::Link::AddParentJoint ( JointPtr *_joint* )

Joints that have this **Link** (p. 542) as a child **Link** (p. 542).

**Parameters**

| | | |
|---|---|---|
| in | *_joint* | **Joint** (p. 496) that is a parent of this link. |

#### 10.96.4.6 virtual void gazebo::physics::Link::AddRelativeForce ( const math::Vector3 & *_force* ) [pure virtual]

Add a force to the body, components are relative to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |

Implemented in **gazebo::physics::SimbodyLink** (p. 904), and **gazebo::physics::DARTLink** (p. 317).

#### 10.96.4.7 virtual void gazebo::physics::Link::AddRelativeTorque ( const math::Vector3 & *_torque* ) [pure virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value to add. |

Implemented in **gazebo::physics::SimbodyLink** (p. 904), and **gazebo::physics::DARTLink** (p. 317).

#### 10.96.4.8 virtual void gazebo::physics::Link::AddTorque ( const math::Vector3 & *_torque* ) [pure virtual]

Add a torque to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value to add to the link. |

Implemented in **gazebo::physics::SimbodyLink** (p. 904), and **gazebo::physics::DARTLink** (p. 318).

#### 10.96.4.9 void gazebo::physics::Link::AttachStaticModel ( ModelPtr & *_model,* const math::Pose & *_offset* )

Attach a static model to this link.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to a static model. |
| in | *_offset* | Pose relative to this link to place the model. |

#### 10.96.4.10 template<typename T > event::ConnectionPtr gazebo::physics::Link::ConnectEnabled ( T *_subscriber* ) [inline]

Connect to the add entity signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Subsciber callback function. |

**Returns**

Pointer to the connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect().

#### 10.96.4.11 void gazebo::physics::Link::DetachAllStaticModels ( )

Detach all static models from this link.

#### 10.96.4.12 void gazebo::physics::Link::DetachStaticModel ( const std::string & *_modelName* )

Detach a static model from this link.

**Parameters**

| | | |
|---|---|---|
| in | *_modelName* | Name of an attached model to detach. |

#### 10.96.4.13 void gazebo::physics::Link::DisconnectEnabled ( event::ConnectionPtr & *_conn* ) [inline]

Disconnect to the add entity signal.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | Connection pointer to disconnect. |

References gazebo::event::EventT< T >::Disconnect().

#### 10.96.4.14 void gazebo::physics::Link::FillMsg ( msgs::Link & *_msg* )

Fill a link message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message to fill |

#### 10.96.4.15 void gazebo::physics::Link::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 382).

Reimplemented in **gazebo::physics::SimbodyLink** (p. 905).

#### 10.96.4.16 double gazebo::physics::Link::GetAngularDamping ( ) const

Get the angular damping factor.

**Returns**

Angular damping.

#### 10.96.4.17 virtual math::Box gazebo::physics::Link::GetBoundingBox ( ) const [virtual]

Get the bounding box for the link and all the child elements.

**Returns**

The link's bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 382).

#### 10.96.4.18 Joint_V gazebo::physics::Link::GetChildJoints ( ) const

Get the child joints.

#### 10.96.4.19 Link_V gazebo::physics::Link::GetChildJointsLinks ( ) const

Returns a vector of children Links connected by joints.

**Returns**

A vector of children Links connected by joints.

#### 10.96.4.20 CollisionPtr gazebo::physics::Link::GetCollision ( const std::string & _name )

Get a child collision by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the collision object. |

**Returns**

Pointer to the collision, NULL if the name was not found.

#### 10.96.4.21 CollisionPtr gazebo::physics::Link::GetCollision ( unsigned int _index ) const

Get a child collision by index.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the collision object. |

**Returns**

Pointer to the collision, NULL if the name was not found.

#### 10.96.4.22 Collision_V gazebo::physics::Link::GetCollisions ( ) const

Get all the child collisions.

**Returns**

A std::vector of all the child collisions.

#### 10.96.4.23 virtual bool gazebo::physics::Link::GetEnabled ( ) const [pure virtual]

Get whether this body is enabled in the physics engine.

**Returns**

True if the link is enabled.

Implemented in **gazebo::physics::DARTLink** (p. 319), and **gazebo::physics::SimbodyLink** (p. 905).

#### 10.96.4.24 virtual bool gazebo::physics::Link::GetGravityMode ( ) const [pure virtual]

Get the gravity mode.

**Returns**

True if gravity is enabled.

Implemented in **gazebo::physics::DARTLink** (p. 319), and **gazebo::physics::SimbodyLink** (p. 905).

#### 10.96.4.25 InertialPtr gazebo::physics::Link::GetInertial ( ) const [inline]

Get the inertia of the link.

**Returns**

Inertia of the link.

References inertial.

#### 10.96.4.26 virtual bool gazebo::physics::Link::GetKinematic ( ) const [inline], [virtual]

Implement this function.

Get whether this body is in the kinematic state.

**Returns**

True if the link is kinematic only.

Reimplemented in **gazebo::physics::DARTLink** (p. 319).

#### 10.96.4.27 double gazebo::physics::Link::GetLinearDamping ( ) const

Get the linear damping factor.

**Returns**

Linear damping.

#### 10.96.4.28 ModelPtr gazebo::physics::Link::GetModel ( ) const

Get the model that this body belongs to.

**Returns**

**Model** (p. 624) that this body belongs to.

#### 10.96.4.29 Joint_V gazebo::physics::Link::GetParentJoints ( ) const

Get the parent joints.

#### 10.96.4.30 Link_V gazebo::physics::Link::GetParentJointsLinks ( ) const

Returns a vector of parent Links connected by joints.

**Returns**

Vector of parent Links connected by joints.

#### 10.96.4.31 math::Vector3 gazebo::physics::Link::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the body.

**Returns**

Angular acceleration of the body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.96.4.32 math::Vector3 gazebo::physics::Link::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the body.

**Returns**

Angular velocity of the body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.96.4.33 math::Vector3 gazebo::physics::Link::GetRelativeForce ( ) const

Get the force applied to the body.

**Returns**

Force applied to the body.

#### 10.96.4.34 math::Vector3 gazebo::physics::Link::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the body.

**Returns**

Linear acceleration of the body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.96.4.35 math::Vector3 gazebo::physics::Link::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the body.

**Returns**

Linear velocity of the body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.96.4.36 math::Vector3 gazebo::physics::Link::GetRelativeTorque ( ) const

Get the torque applied to the body.

**Returns**

Torque applied to the body.

#### 10.96.4.37 bool gazebo::physics::Link::GetSelfCollide ( ) const

Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.

**Returns**

True if self collision is enabled.

#### 10.96.4.38 unsigned int gazebo::physics::Link::GetSensorCount ( ) const

Get sensor count.

This will return the number of sensors created by the link when it was loaded. This function is commonly used with **Link::GetSensorName** (p. 554).

**Returns**

The number of sensors created by the link.

#### 10.96.4.39 std::string gazebo::physics::Link::GetSensorName ( unsigned int *_index* ) const

Get sensor name.

Get the name of a sensor based on an index. The index should be in the range of 0...**Link::GetSensorCount()** (p. 554).

**Note**

A **Link** (p. 542) does not manage or maintain a pointer to a **sensors::Sensor** (p. 837). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 850). This was done to separate the physics engine from the sensor engine.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the sensor name. |

**Returns**

The name of the sensor, or empty string if the index is out of bounds.

#### 10.96.4.40 math::Vector3 gazebo::physics::Link::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the body in the world frame.

**Returns**

Angular acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.96.4.41 virtual math::Vector3 gazebo::physics::Link::GetWorldCoGLinearVel ( ) const [pure virtual]

Get the linear velocity at the body's center of gravity in the world frame.

**Returns**

Linear velocity at the body's center of gravity in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 319), and **gazebo::physics::SimbodyLink** (p. 905).

#### 10.96.4.42 math::Pose gazebo::physics::Link::GetWorldCoGPose ( ) const

Get the pose of the body's center of gravity in the world coordinate frame.

**Returns**

Pose of the body's center of gravity in the world coordinate frame.

#### 10.96.4.43 virtual math::Vector3 gazebo::physics::Link::GetWorldForce ( ) const [pure virtual]

Get the force applied to the body in the world frame.

**Returns**

Force applied to the body in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 320), and **gazebo::physics::SimbodyLink** (p. 906).

#### 10.96.4.44 math::Vector3 gazebo::physics::Link::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the body in the world frame.

**Returns**

Linear acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.96.4.45 virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel ( const math::Vector3 & *_offset* = math::Vector3(0, 0, 0) ) const [pure virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.

If no offset is given, the velocity at the origin of the **Link** (p. 542) frame will be returned.

**Parameters**

| | | |
|---|---|---|
| in | *_offset* | Offset of the point from the origin of the **Link** (p. 542) frame, expressed in the body-fixed frame. |

**Returns**

Linear velocity of the point on the body

Implemented in **gazebo::physics::DARTLink** (p. 320), and **gazebo::physics::SimbodyLink** (p. 906).

#### 10.96.4.46 virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel ( const math::Vector3 & *_offset*, const math::Quaternion & *_q* ) const [pure virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

**Parameters**

| | | |
|---|---|---|
| in | *_offset* | Offset from the origin of the link frame expressed in a frame defined by _q. |
| in | *_q* | Describes the rotation of a reference frame relative to the world reference frame. |

**Returns**

Linear velocity of the point on the body in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 320), and **gazebo::physics::SimbodyLink** (p. 906).

#### 10.96.4.47 virtual math::Vector3 gazebo::physics::Link::GetWorldTorque ( ) const [pure virtual]

Get the torque applied to the body in the world frame.

**Returns**

Torque applied to the body in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 320), and **gazebo::physics::SimbodyLink** (p. 906).

#### 10.96.4.48 virtual void gazebo::physics::Link::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 167).

Reimplemented in **gazebo::physics::DARTLink** (p. 321), and **gazebo::physics::SimbodyLink** (p. 907).

#### 10.96.4.49 virtual void gazebo::physics::Link::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the body based on an SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters. |

Reimplemented from **gazebo::physics::Entity** (p. 386).

Reimplemented in **gazebo::physics::DARTLink** (p. 321), and **gazebo::physics::SimbodyLink** (p. 907).

#### 10.96.4.50 virtual void gazebo::physics::Link::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **gazebo::physics::Entity** (p. 386).

Reimplemented in **gazebo::physics::DARTLink** (p. 321), and **gazebo::physics::SimbodyLink** (p. 907).

#### 10.96.4.51 void gazebo::physics::Link::ProcessMsg ( const msgs::Link & *_msg* )

Update parameters from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to read. |

#### 10.96.4.52 virtual void gazebo::physics::Link::RemoveChild ( EntityPtr *_child* ) [virtual]

#### 10.96.4.53 void gazebo::physics::Link::RemoveChildJoint ( const std::string & *_jointName* )

Remove Joints that have this **Link** (p. 542) as a parent **Link** (p. 542).

**Parameters**

| | | |
|---|---|---|
| in | *_jointName* | Child **Joint** (p. 496) name. |

#### 10.96.4.54 void gazebo::physics::Link::RemoveCollision ( const std::string & *_name* )

Remove a collision from the link.

**Parameters**

| | |
|---|---|
| *int]* | _name Name of the collision to remove. |

#### 10.96.4.55 void gazebo::physics::Link::RemoveParentJoint ( const std::string & *_jointName* )

Remove Joints that have this **Link** (p. 542) as a child **Link** (p. 542).

**Parameters**

| | | |
|---|---|---|
| in | *_jointName* | Parent **Joint** (p. 496) name. |

#### 10.96.4.56 void gazebo::physics::Link::Reset ( ) [virtual]

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 387).

#### 10.96.4.57 void gazebo::physics::Link::ResetPhysicsStates ( )

Reset the link.

#### 10.96.4.58 void gazebo::physics::Link::SetAngularAccel ( const math::Vector3 & *_accel* )

Set the angular acceleration of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_accel* | Angular acceleration. |

#### 10.96.4.59 virtual void gazebo::physics::Link::SetAngularDamping ( double *_damping* ) [pure virtual]

Set the angular damping factor.

**Parameters**

| | | |
|---|---|---|
| in | *_damping* | Angular damping factor. |

Implemented in **gazebo::physics::DARTLink** (p. 321), and **gazebo::physics::SimbodyLink** (p. 907).

#### 10.96.4.60 virtual void gazebo::physics::Link::SetAngularVel ( const math::Vector3 & *_vel* ) [pure virtual]

Set the angular velocity of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Angular velocity. |

Implemented in **gazebo::physics::DARTLink** (p. 321), and **gazebo::physics::SimbodyLink** (p. 907).

#### 10.96.4.61 virtual void gazebo::physics::Link::SetAutoDisable ( bool *_disable* ) [pure virtual]

Allow the link to auto disable.

**Parameters**

| | | |
|---|---|---|
| in | *_disable* | If true, the link is allowed to auto disable. |

Implemented in **gazebo::physics::DARTLink** (p. 321), and **gazebo::physics::SimbodyLink** (p. 907).

#### 10.96.4.62 void gazebo::physics::Link::SetCollideMode ( const std::string & *_mode* )

Set the collide mode of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_mode* | **Collision** (p. 220) Mode, this can be: [all\|none\|sensors\|fixed\|ghost] all: collides with everything none: collides with nothing sensors: collides with everything else but other sensors fixed: collides with everything else but other fixed ghost: collides with everything else but other ghost |

#### 10.96.4.63 virtual void gazebo::physics::Link::SetEnabled ( bool *_enable* ) const [pure virtual]

Set whether this body is enabled.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | True to enable the link in the physics engine. |

Implemented in **gazebo::physics::DARTLink** (p. 322), and **gazebo::physics::SimbodyLink** (p. 908).

#### 10.96.4.64 virtual void gazebo::physics::Link::SetForce ( const math::Vector3 & *_force* ) [pure virtual]

Set the force applied to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force value. |

Implemented in **gazebo::physics::DARTLink** (p. 322), and **gazebo::physics::SimbodyLink** (p. 908).

#### 10.96.4.65 virtual void gazebo::physics::Link::SetGravityMode ( bool *_mode* ) [pure virtual]

Set whether gravity affects this body.

**Parameters**

| | | |
|---|---|---|
| in | *_mode* | True to enable gravity. |

Implemented in **gazebo::physics::DARTLink** (p. 322), and **gazebo::physics::SimbodyLink** (p. 908).

#### 10.96.4.66 void gazebo::physics::Link::SetInertial ( const InertialPtr & *_inertial* )

Set the mass of the link.

[in] _inertial **Inertial** (p. 483) value for the link.

#### 10.96.4.67 virtual void gazebo::physics::Link::SetKinematic ( const bool & *_kinematic* ) [virtual]

Implement this function.

Set whether this body is in the kinematic state.

**Parameters**

| | | |
|---|---|---|
| in | *_kinematic* | True to make the link kinematic only. |

Reimplemented in **gazebo::physics::DARTLink** (p. 322).

#### 10.96.4.68 void gazebo::physics::Link::SetLaserRetro ( float *_retro* )

Set the laser retro reflectiveness.

**Parameters**

| | | |
|---|---|---|
| in | *_retro* | Retro value for all child collisions. |

#### 10.96.4.69 void gazebo::physics::Link::SetLinearAccel ( const **math::Vector3** & *_accel* )

Set the linear acceleration of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_accel* | Linear acceleration. |

#### 10.96.4.70 virtual void gazebo::physics::Link::SetLinearDamping ( double *_damping* ) [pure virtual]

Set the linear damping factor.

**Parameters**

| | | |
|---|---|---|
| in | *_damping* | Linear damping factor. |

Implemented in **gazebo::physics::DARTLink** (p. 323), and **gazebo::physics::SimbodyLink** (p. 908).

#### 10.96.4.71 virtual void gazebo::physics::Link::SetLinearVel ( const **math::Vector3** & *_vel* ) [pure virtual]

Set the linear velocity of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Linear velocity. |

Implemented in **gazebo::physics::DARTLink** (p. 323), and **gazebo::physics::SimbodyLink** (p. 908).

#### 10.96.4.72 virtual void gazebo::physics::Link::SetLinkStatic ( bool *_static* ) [pure virtual]

Freeze link to ground (inertial frame).

**Parameters**

| | | |
|---|---|---|
| in | *_static* | if true, freeze link to ground. Otherwise unfreeze link. |

Implemented in **gazebo::physics::SimbodyLink** (p. 909), and **gazebo::physics::DARTLink** (p. 323).

#### 10.96.4.73 void gazebo::physics::Link::SetPublishData ( bool _enable )

Enable/Disable link data publishing.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | True to enable publishing, false to stop publishing |

#### 10.96.4.74 void gazebo::physics::Link::SetScale ( const math::Vector3 & _scale )

Set the scale of the link.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the link to. |

#### 10.96.4.75 virtual bool gazebo::physics::Link::SetSelected ( bool _set ) [virtual]

Set whether this entity has been selected by the user through the gui.

**Parameters**

| | | |
|---|---|---|
| in | *_set* | True to set the link as selected. |

Reimplemented from **gazebo::physics::Base** (p. 170).

#### 10.96.4.76 virtual void gazebo::physics::Link::SetSelfCollide ( bool _collide ) [pure virtual]

Set whether this body will collide with others in the model.

**Parameters**

| | | |
|---|---|---|
| in | *_collid* | True to enable collisions. |

Implemented in **gazebo::physics::DARTLink** (p. 323), and **gazebo::physics::SimbodyLink** (p. 909).

#### 10.96.4.77 void gazebo::physics::Link::SetState ( const LinkState & _state )

Set the current link state.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | The state to set the link to. |

#### 10.96.4.78 virtual void gazebo::physics::Link::SetTorque ( const math::Vector3 & _torque ) [pure virtual]

Set the torque applied to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value. |

Implemented in **gazebo::physics::DARTLink** (p. 323), and **gazebo::physics::SimbodyLink** (p. 909).

#### 10.96.4.79 void gazebo::physics::Link::Update ( const common::UpdateInfo & *_info* )

Update the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_info* | Update information. |

#### 10.96.4.80 virtual void gazebo::physics::Link::UpdateMass ( ) [inline], [virtual]

Update the mass matrix.

#### 10.96.4.81 virtual void gazebo::physics::Link::UpdateParameters ( sdf::ElementPtr *_sdf* ) [virtual]

Update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::Entity** (p. 389).

#### 10.96.4.82 virtual void gazebo::physics::Link::UpdateSurface ( ) [inline], [virtual]

Update surface parameters.

### 10.96.5 Member Data Documentation

#### 10.96.5.1 math::Vector3 gazebo::physics::Link::angularAccel [protected]

Angular acceleration.

#### 10.96.5.2 std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset [protected]

Offsets for the attached models.

#### 10.96.5.3 std::vector<std::string> gazebo::physics::Link::cgVisuals [protected]

Center of gravity visual elements.

#### 10.96.5.4 InertialPtr gazebo::physics::Link::inertial [protected]

**Inertial** (p. 483) properties.

Referenced by GetInertial().

#### 10.96.5.5 math::Vector3 gazebo::physics::Link::linearAccel [protected]

Linear acceleration.

#### 10.96.5.6 Visuals_M gazebo::physics::Link::visuals [protected]

**Link** (p. 542) visual elements.

The documentation for this class was generated from the following file:

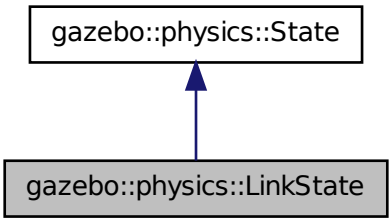
- **Link.hh**

## 10.97 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 542) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::LinkState:




### Public Member Functions

- **LinkState** ()

  *Default constructor.*
- **LinkState** (const **LinkPtr** _link, const **common::Time** &_realTime, const **common::Time** &_simTime)

  *Constructor.*
- **LinkState** (const **LinkPtr** _link)

  *Constructor.*
- **LinkState** (const sdf::ElementPtr _sdf)

*Constructor.*

- virtual ~**LinkState** ()

  *Destructor.*

- void **FillSDF** (sdf::ElementPtr _sdf)

  *Populate a state SDF element with data from the object.*

- const **math::Pose** & **GetAcceleration** () const

  *Get the link acceleration.*

- **CollisionState GetCollisionState** (unsigned int _index) const

  *Get a collision state.*

- **CollisionState GetCollisionState** (const std::string &_collisionName) const

  *Get a link state by link name.*

- unsigned int **GetCollisionStateCount** () const

  *Get the number of link states.*

- const std::vector < **CollisionState** > & **GetCollisionStates** () const

  *Get the collision states.*

- const **math::Pose** & **GetPose** () const

  *Get the link pose.*

- const **math::Pose** & **GetVelocity** () const

  *Get the link velocity.*

- const **math::Pose** & **GetWrench** () const

  *Get the force applied to the **Link** (p. 542).*

- bool **IsZero** () const

  *Return true if the values in the state are zero.*

- void **Load** (const **LinkPtr** _link, const **common::Time** &_realTime, const **common::Time** &_simTime)

  *Load a **LinkState** (p. 563) from a **Link** (p. 542) pointer.*

- virtual void **Load** (const sdf::ElementPtr _elem)

  *Load state from SDF element.*

- **LinkState operator+** (const **LinkState** &_state) const

  *Addition operator.*

- **LinkState operator-** (const **LinkState** &_state) const

  *Subtraction operator.*

- **LinkState** & **operator=** (const **LinkState** &_state)

  *Assignment operator.*

- virtual void **SetRealTime** (const **common::Time** &_time)

  *Set the real time when this state was generated.*

- virtual void **SetSimTime** (const **common::Time** &_time)

  *Set the sim time when this state was generated.*

- virtual void **SetWallTime** (const **common::Time** &_time)

  *Set the wall time when this state was generated.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::LinkState** &_state)

  *Stream insertion operator.*

## Additional Inherited Members

### 10.97.1 Detailed Description

Store state information of a **physics::Link** (p. 542) object.

This class captures the entire state of a **Link** (p. 542) at one specific time during a simulation run.

**State** (p. 998) of a **Link** (p. 542) includes the state of itself all its child **Collision** (p. 220) entities.

### 10.97.2 Constructor & Destructor Documentation

#### 10.97.2.1 gazebo::physics::LinkState::LinkState ( )

Default constructor.

#### 10.97.2.2 gazebo::physics::LinkState::LinkState ( const LinkPtr *_link*, const common::Time & *_realTime*, const common::Time & *_simTime* )

Constructor.

Build a **LinkState** (p. 563) from an existing **Link** (p. 542).

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the **Link** (p. 542) from which to gather state info. |
| in | *_realTime* | Real time stamp. |
| in | *_simTime* | Sim time stamp |

#### 10.97.2.3 gazebo::physics::LinkState::LinkState ( const LinkPtr *_link* ) [explicit]

Constructor.

Build a **LinkState** (p. 563) from an existing **Link** (p. 542).

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the **Link** (p. 542) from which to gather state info. |

#### 10.97.2.4 gazebo::physics::LinkState::LinkState ( const sdf::ElementPtr *_sdf* ) [explicit]

Constructor.

Build a **LinkState** (p. 563) from SDF data

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF data to load a link state from. |

#### 10.97.2.5 virtual gazebo::physics::LinkState::~LinkState ( ) [virtual]

Destructor.

### 10.97.3 Member Function Documentation

#### 10.97.3.1 void gazebo::physics::LinkState::FillSDF ( sdf::ElementPtr _sdf )

Populate a state SDF element with data from the object.

**Parameters**

| | | |
|---|---|---|
| out | *_sdf* | SDF element to populate. |

#### 10.97.3.2 const math::Pose& gazebo::physics::LinkState::GetAcceleration ( ) const

Get the link acceleration.

**Returns**

The acceleration represented as a **math::Pose** (p. 734).

#### 10.97.3.3 CollisionState gazebo::physics::LinkState::GetCollisionState ( unsigned int _index ) const

Get a collision state.

Get a **Collision** (p. 220) **State** (p. 998) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 567).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the **CollisionState** (p. 229). |

**Returns**

**State** (p. 998) of the **Collision** (p. 220).

**Exceptions**

| | |
|---|---|
| ***common::Exception*** *(p. 416)* | When _index is invalid. |

#### 10.97.3.4 CollisionState gazebo::physics::LinkState::GetCollisionState ( const std::string & _collisionName ) const

Get a link state by link name.

Searches through all CollisionStates. Returns the **CollisionState** (p. 229) with the matching name, if any.

**Parameters**

| | | |
|---|---|---|
| in | *_collisionName* | Name of the **CollisionState** (p. 229) |

**Returns**

**State** (p. 998) of the **Collision** (p. 220).

**Exceptions**

| | |
|---|---|
| ***common::Exception*** *(p. 416)* | When _collisionName is invalid |

#### 10.97.3.5 unsigned int gazebo::physics::LinkState::GetCollisionStateCount ( ) const

Get the number of link states.

This returns the number of Collisions recorded.

**Returns**

Number of **CollisionState** (p. 229) recorded.

#### 10.97.3.6 const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates ( ) const

Get the collision states.

**Returns**

A vector of collision states.

#### 10.97.3.7 const math::Pose& gazebo::physics::LinkState::GetPose ( ) const

Get the link pose.

**Returns**

The **math::Pose** (p. 734) of the **Link** (p. 542).

#### 10.97.3.8 const math::Pose& gazebo::physics::LinkState::GetVelocity ( ) const

Get the link velocity.

**Returns**

The velocity represented as a **math::Pose** (p. 734).

#### 10.97.3.9 const math::Pose& gazebo::physics::LinkState::GetWrench ( ) const

Get the force applied to the **Link** (p. 542).

**Returns**

Magnitude of the force.

#### 10.97.3.10 bool gazebo::physics::LinkState::IsZero ( ) const

Return true if the values in the state are zero.

**Returns**

True if the values in the state are zero.

#### 10.97.3.11 void gazebo::physics::LinkState::Load ( const LinkPtr *_link,* const common::Time & *_realTime,* const common::Time & *_simTime* )

Load a **LinkState** (p. 563) from a **Link** (p. 542) pointer.

Build a **LinkState** (p. 563) from an existing **Link** (p. 542).

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the **Link** (p. 542) from which to gather state info. |
| in | *_realTime* | Real time stamp. |
| in | *_simTime* | Sim time stamp |

#### 10.97.3.12 virtual void gazebo::physics::LinkState::Load ( const sdf::ElementPtr *_elem* ) [virtual]

Load state from SDF element.

Load **LinkState** (p. 563) information from stored data in and SDF::Element.

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | Pointer to the SDF::Element containing state info. |

Reimplemented from **gazebo::physics::State** (p. 1000).

#### 10.97.3.13 LinkState gazebo::physics::LinkState::operator+ ( const LinkState & *_state* ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to add. |

**Returns**

The resulting state.

#### 10.97.3.14 LinkState gazebo::physics::LinkState::operator- ( const LinkState & *_state* ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to substract. |

**Returns**

The resulting state.

#### 10.97.3.15 LinkState& gazebo::physics::LinkState::operator= ( const LinkState & *_state* )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **State** (p. 998) value |

**Returns**

this

#### 10.97.3.16 virtual void gazebo::physics::LinkState::SetRealTime ( const common::Time & *_time* ) [virtual]

Set the real time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Clock time since simulation was stated. |

Reimplemented from **gazebo::physics::State** (p. 1001).

#### 10.97.3.17 virtual void gazebo::physics::LinkState::SetSimTime ( const common::Time & *_time* ) [virtual]

Set the sim time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Simulation time when the data was recorded. |

Reimplemented from **gazebo::physics::State** (p. 1001).

#### 10.97.3.18 virtual void gazebo::physics::LinkState::SetWallTime ( const common::Time & *_time* ) [virtual]

Set the wall time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The absolute clock time when the **State** (p. 998) data was recorded. |

Reimplemented from **gazebo::physics::State** (p. 1002).

### 10.97.4 Friends And Related Function Documentation

#### 10.97.4.1 std::ostream& operator<< ( std::ostream & *_out,* const gazebo::physics::LinkState & *_state* ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_state* | **Link** (p. 542) state to output |

**Returns**

the stream

Disabling this for efficiency.

Disabling this for efficiency.

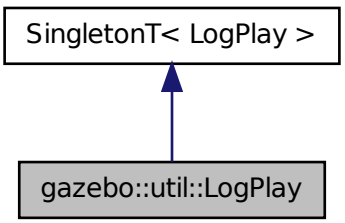
The documentation for this class was generated from the following file:

- **LinkState.hh**

## 10.98 gazebo::util::LogPlay Class Reference

`#include <LogPlay.hh>`

Inheritance diagram for gazebo::util::LogPlay:



## Public Member Functions

- bool **GetChunk** (unsigned int _index, std::string &_data)

  *Get data for a particular chunk index.*
- unsigned int **GetChunkCount** () const

  *Get the number of chunks (steps) in the open log file.*
- std::string **GetEncoding** () const

  *Get the type of encoding used for current chunck in the open log file.*
- std::string **GetGazeboVersion** () const

  *Get the Gazebo version number of the open log file.*
- std::string **GetHeader** () const

  *Get the header that was read from a log file.*
- std::string **GetLogVersion** () const

  *Get the log version number of the open log file.*
- uint32_t **GetRandSeed** () const

  *Get the random number seed of the open log file.*
- bool **IsOpen** () const

  *Return true if a file is open.*
- void **Open** (const std::string &_logFile)

  *Open a log file for reading.*
- bool **Step** (std::string &_data)

  *Step through the open log file.*

## Additional Inherited Members

## 10.98.1 Member Function Documentation

### 10.98.1.1 bool gazebo::util::LogPlay::GetChunk ( unsigned int *_index*, std::string & *_data* )

Get data for a particular chunk index.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the chunk. |
| out | *_data* | Storage for the chunk's data. |

**Returns**

True if the _index was valid.

### 10.98.1.2 unsigned int gazebo::util::LogPlay::GetChunkCount ( ) const

Get the number of chunks (steps) in the open log file.

**Returns**

The number of recorded states in the log file.

#### 10.98.1.3 std::string gazebo::util::LogPlay::GetEncoding ( ) const

Get the type of encoding used for current chunck in the open log file.

**Returns**

The type of encoding. An empty string will be returned if **LogPlay::Step** (p. 573) has not been called at least once.

#### 10.98.1.4 std::string gazebo::util::LogPlay::GetGazeboVersion ( ) const

Get the Gazebo version number of the open log file.

**Returns**

The Gazebo version of the open log file. Empty string if a log file is not open.

#### 10.98.1.5 std::string gazebo::util::LogPlay::GetHeader ( ) const

Get the header that was read from a log file.

Should call **LogPlay::Open** (p. 573) first.

**Returns**

Header of the open log file.

#### 10.98.1.6 std::string gazebo::util::LogPlay::GetLogVersion ( ) const

Get the log version number of the open log file.

**Returns**

The log version of the open log file. Empty string if a log file is not open.

#### 10.98.1.7 uint32_t gazebo::util::LogPlay::GetRandSeed ( ) const

Get the random number seed of the open log file.

**Returns**

The random number seed the open log file. The current random number seed, as defined in **math::Rand::GetSeed** (p. 776).

#### 10.98.1.8 bool gazebo::util::LogPlay::IsOpen ( ) const

Return true if a file is open.

**Returns**

True if a log file is open.

#### 10.98.1.9 void gazebo::util::LogPlay::Open ( const std::string & _logFile )

Open a log file for reading.

Open a log file that was previously recorded.

**Parameters**

| | | |
|---|---|---|
| in | *_logFile* | The file to load |

**Exceptions**

| | |
|---|---|
| *Exception* | |

#### 10.98.1.10 bool gazebo::util::LogPlay::Step ( std::string & _data )

Step through the open log file.

**Parameters**

| | | |
|---|---|---|
| out | *_data* | Data from next entry in the log file. |

The documentation for this class was generated from the following file:

- **LogPlay.hh**

## 10.99 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

### 10.99.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 573) to open a log file (Logplay::Open), and access the recorded state information. Iterators are available to step through the state information. It is also possible to replay the data in a World using the Play functions. Replay involves reading and applying state information to a World.

**See Also**

LogRecord, State

The documentation for this class was generated from the following file:

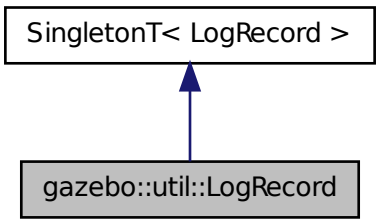
- **LogPlay.hh**

## 10.100 gazebo::util::LogRecord Class Reference

addtogroup gazebo_util

`#include <util/util.hh>`

Inheritance diagram for gazebo::util::LogRecord:




## Public Member Functions

- void **Add** (const std::string &_name, const std::string &_filename, boost::function< bool(std::ostringstream &)> _logCallback)

  *Add an object to a log file.*
- void **Fini** ()

  *Finialize, and shutdown.*
- std::string **GetBasePath** () const

  *Get the base path for a log recording.*
- unsigned int **GetBufferSize** () const

  *Get the size of the buffer.*
- const std::string & **GetEncoding** () const

  *Get the encoding used.*
- std::string **GetFilename** (const std::string &_name="") const

  *Get the filename for a log object.*
- unsigned int **GetFileSize** (const std::string &_name="") const

  *Get the file size for a log object.*
- bool **GetFirstUpdate** () const

  *Return true if an Update has not yet been completed.*
- bool **GetPaused** () const

  *Get whether logging is paused.*
- bool **GetRunning** () const

  *Get whether logging is running.*
- **common::Time GetRunTime** () const

  *Get the run time in sim time.*
- bool **Init** (const std::string &_subdir)

  *Initialize logging into a subdirectory.*
- bool **IsReadyToStart** () const

  *Get whether the logger is ready to start, which implies that any previous runs have finished.*
- void **Notify** ()

*Tell the recorder that an update should occur.*

- bool **Remove** (const std::string &_name)

  *Remove an entity from a log.*
- void **SetBasePath** (const std::string &_path)

  *Set the base path.*
- void **SetPaused** (bool _paused)

  *Set whether logging should pause.*
- bool **Start** (const std::string &_encoding="zlib", const std::string &_path="")

  *Start the logger.*
- void **Stop** ()

  *Stop the logger.*
- void **Write** (bool _force=false)

  *Write all logs.*

## Additional Inherited Members

### 10.100.1 Detailed Description

addtogroup gazebo_util

Handles logging of data to disk

The **LogRecord** (p. 573) class is a Singleton that manages data logging of any entity within a running simulation. An entity may be a World, Model, or any of their child entities. This class only writes log files, see **LogPlay** (p. 570) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 575) function, and stopped through the **LogRecord::Remove** (p. 578) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 575) function.

The **LogRecord** (p. 573) is updated at the start of each simulation step. This guarantees that all data is stored.

**See Also**

**Logplay** (p. 573), State

### 10.100.2 Member Function Documentation

#### 10.100.2.1 void gazebo::util::LogRecord::Add ( const std::string & *_name,* const std::string & *_filename,* boost::function< bool(std::ostringstream &)> *_logCallback* )

Add an object to a log file.

Add a new object to a log. An object can be any valid named object in simulation, including the world itself. Duplicate additions are ignored. Objects can be added to the same file by specifying the same _filename.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the object to log. |
| in | *_filename* | Filename of the log file. |
| in | *_logCallback* | Function used to log data for the object. Typically an object will have a log function that outputs data to the provided ofstream. |

**Exceptions**

| | |
|---|---|
| *Exception* | |

#### 10.100.2.2 void gazebo::util::LogRecord::Fini ( )

Finialize, and shutdown.

#### 10.100.2.3 std::string gazebo::util::LogRecord::GetBasePath ( ) const

Get the base path for a log recording.

**Returns**

Path for log recording.

#### 10.100.2.4 unsigned int gazebo::util::LogRecord::GetBufferSize ( ) const

Get the size of the buffer.

**Returns**

Size of the buffer, in bytes.

#### 10.100.2.5 const std::string& gazebo::util::LogRecord::GetEncoding ( ) const

Get the encoding used.

**Returns**

Either [txt, zlib, or bz2], where txt is plain txt and bz2 and zlib are compressed data with Base64 encoding.

#### 10.100.2.6 std::string gazebo::util::LogRecord::GetFilename ( const std::string & *_name* = " " ) const

Get the filename for a log object.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the log object. |

**Returns**

Filename, empty string if not found.

#### 10.100.2.7 unsigned int gazebo::util::LogRecord::GetFileSize ( const std::string & *_name* = " " ) const

Get the file size for a log object.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the log object. |

**Returns**

Size in bytes.

#### 10.100.2.8 bool gazebo::util::LogRecord::GetFirstUpdate ( ) const

Return true if an Update has not yet been completed.

**Returns**

True if an Update has not yet been completed.

#### 10.100.2.9 bool gazebo::util::LogRecord::GetPaused ( ) const

Get whether logging is paused.

**Returns**

True if logging is paused.

**See Also**

**LogRecord::SetPaused** (p. 578)

#### 10.100.2.10 bool gazebo::util::LogRecord::GetRunning ( ) const

Get whether logging is running.

**Returns**

True if logging has been started.

#### 10.100.2.11 common::Time gazebo::util::LogRecord::GetRunTime ( ) const

Get the run time in sim time.

**Returns**

Run sim time.

#### 10.100.2.12 bool gazebo::util::LogRecord::Init ( const std::string & *_subdir* )

Initialize logging into a subdirectory.

Init may only be called once, False will be returned if called multiple times.

**Parameters**

| | | |
|---|---|---|
| in | *_subdir* | Directory to record to |

**Returns**

True if successful.

#### 10.100.2.13 bool gazebo::util::LogRecord::IsReadyToStart ( ) const

Get whether the logger is ready to start, which implies that any previous runs have finished.

#### 10.100.2.14 void gazebo::util::LogRecord::Notify ( )

Tell the recorder that an update should occur.

#### 10.100.2.15 bool gazebo::util::LogRecord::Remove ( const std::string & *_name* )

Remove an entity from a log.

Removes an entity from the logger. The stops data recording for the entity and all its children. For example, specifying a world will stop all data logging.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the log |

**Returns**

True if the entity existed and was removed. False if the entity was not registered with the logger.

#### 10.100.2.16 void gazebo::util::LogRecord::SetBasePath ( const std::string & *_path* )

Set the base path.

**Parameters**

| | | |
|---|---|---|
| in | *_path* | Path to the new logging location. |

#### 10.100.2.17 void gazebo::util::LogRecord::SetPaused ( bool *_paused* )

Set whether logging should pause.

A paused state means the log file is still open, but data is not written to it.

**Parameters**

| | | |
|---|---|---|
| in | *_paused* | True to pause data logging. |

**See Also**

**LogRecord::GetPaused** (p. 577)

#### 10.100.2.18 bool gazebo::util::LogRecord::Start ( const std::string & *_encoding* = "zlib", const std::string & *_path* = "" )

Start the logger.

**Parameters**

| | | |
|---|---|---|
| in | *_encoding* | The type of encoding (txt, zlib, or bz2). |
| in | *_path* | Path in which to store log files. |

#### 10.100.2.19 void gazebo::util::LogRecord::Stop ( )

Stop the logger.

#### 10.100.2.20 void gazebo::util::LogRecord::Write ( bool *_force* = false )

Write all logs.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True to skip waiting on dataAvailableCondition. |

The documentation for this class was generated from the following file:

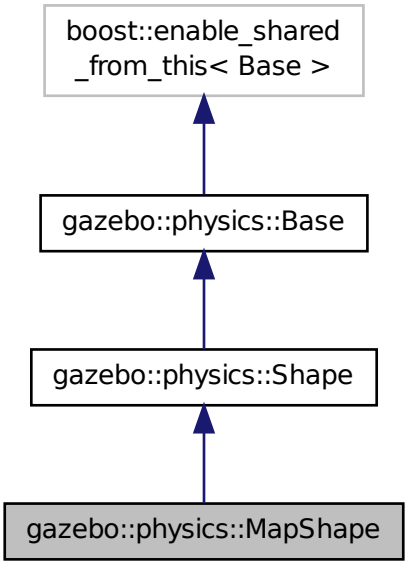
- **LogRecord.hh**

## 10.101 gazebo::physics::MapShape Class Reference

Creates box extrusions based on an image.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::MapShape:




## Public Member Functions

- **MapShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**MapShape** ()

  *Destructor.*
- void **FillMsg** (msgs::Geometry &_msg)

  *Fills out a msgs::Geometry message containing information about this map geometry object.*
- int **GetGranularity** () const

  *Returns granularity of this geometry.*
- double **GetHeight** () const

  *Returns height of this geometry.*
- virtual **math::Vector3 GetScale** () const

  *Returns scaling factor for this geometry.*
- int **GetThreshold** () const

  *Returns image threshold for this geometry.*
- std::string **GetURI** () const

  *Returns the image URI for this geometry.*
- virtual void **Init** ()

  *Init the map.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the map.*

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *: Implement this function.*
- void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the map shape.*
- void **Update** ()

  *Update function.*

### Additional Inherited Members

### 10.101.1 Detailed Description

Creates box extrusions based on an image.

This function is not yet complete, to be implemented.

### 10.101.2 Constructor & Destructor Documentation

#### 10.101.2.1 gazebo::physics::MapShape::MapShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent collision object. |

#### 10.101.2.2 virtual gazebo::physics::MapShape::~MapShape ( ) [virtual]

Destructor.

### 10.101.3 Member Function Documentation

#### 10.101.3.1 void gazebo::physics::MapShape::FillMsg ( msgs::Geometry & *_msg* ) [virtual]

Fills out a msgs::Geometry message containing information about this map geometry object.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to fill with this object's data. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.101.3.2 int gazebo::physics::MapShape::GetGranularity ( ) const

Returns granularity of this geometry.

**Returns**

Granularity (amount of error betweent the image pixels and the 3D shapes created).

#### 10.101.3.3 double gazebo::physics::MapShape::GetHeight ( ) const

Returns height of this geometry.

All regions in image with value larger than **MapShape::scale** (p. 865) will be replaced by boxes with MapShape::height.

**Returns**

Height of the map shapes.

#### 10.101.3.4 virtual math::Vector3 gazebo::physics::MapShape::GetScale ( ) const [virtual]

Returns scaling factor for this geometry.

**Returns**

Scaling factor.

Reimplemented from **gazebo::physics::Shape** (p. 863).

#### 10.101.3.5 int gazebo::physics::MapShape::GetThreshold ( ) const

Returns image threshold for this geometry.

All regions in image with value larger than **MapShape::scale** (p. 865) will be replaced by boxes with MapShape::height.

**Returns**

Image threshold value.

#### 10.101.3.6 std::string gazebo::physics::MapShape::GetURI ( ) const

Returns the image URI for this geometry.

**Returns**

The image URI that was used to load the map.

#### 10.101.3.7 virtual void gazebo::physics::MapShape::Init ( ) [virtual]

Init the map.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.101.3.8 virtual void gazebo::physics::MapShape::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the map.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Load the map from SDF values. |

Reimplemented from **gazebo::physics::Base** (p. 168).

#### 10.101.3.9 virtual void gazebo::physics::MapShape::ProcessMsg ( const msgs::Geometry & *_msg* ) [virtual]

: Implement this function.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to process, which will alter the map. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.101.3.10 void gazebo::physics::MapShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set the scale of the map shape.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the map shape to. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.101.3.11 void gazebo::physics::MapShape::Update ( ) [virtual]

Update function.

Reimplemented from **gazebo::physics::Base** (p. 171).

The documentation for this class was generated from the following file:

- **MapShape.hh**

## 10.102 gazebo::Master Class Reference

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

`#include <gazebo_core.hh>`

### Public Member Functions

- **Master** ()
  *Constructor.*
- virtual ~**Master** ()
  *Destructor.*
- void **Fini** ()
  *Finalize the master.*
- void **Init** (uint16_t _port)
  *Initialize.*
- void **Run** ()

*Run the master.*

- void **RunOnce** ()

  *Run the master one iteration.*

- void **RunThread** ()

  *Run the master in a new thread.*

- void **Stop** ()

  *Stop the master.*

### 10.102.1 Detailed Description

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

Base class for simulation server that handles commandline options, starts a **Master** (p. 583), runs World update and sensor generation loops.

### 10.102.2 Constructor & Destructor Documentation

#### 10.102.2.1 gazebo::Master::Master ( )

Constructor.

#### 10.102.2.2 virtual gazebo::Master::~Master ( ) [virtual]

Destructor.

### 10.102.3 Member Function Documentation

#### 10.102.3.1 void gazebo::Master::Fini ( )

Finalize the master.

#### 10.102.3.2 void gazebo::Master::Init ( uint16_t *_port* )

Initialize.

**Parameters**

| | | |
|---|---|---|
| in | *_port* | The master's port |

#### 10.102.3.3 void gazebo::Master::Run ( )

Run the master.

#### 10.102.3.4 void gazebo::Master::RunOnce ( )

Run the master one iteration.

#### 10.102.3.5 void gazebo::Master::RunThread ( )

Run the master in a new thread.

#### 10.102.3.6 void gazebo::Master::Stop ( )

Stop the master.

The documentation for this class was generated from the following file:

- **Master.hh**

## 10.103 gazebo::common::Material Class Reference

Encapsulates description of a material.

`#include <common/common.hh>`

### Public Types

- enum **BlendMode** { **ADD**, **MODULATE**, **REPLACE**, **BLEND_COUNT** }
- enum **ShadeMode** {
  **FLAT**, **GOURAUD**, **PHONG**, **BLINN**,
  **SHADE_COUNT** }

### Public Member Functions

- **Material** ()
  *Constructor.*
- **Material** (const **Color** &_clr)
  *Create a material with a default color.*
- virtual ~**Material** ()
  *Destructor.*
- **Color GetAmbient** () const
  *Get the ambient color.*
- void **GetBlendFactors** (double &_srcFactor, double &_dstFactor)
  *Get the blend factors.*
- **BlendMode GetBlendMode** () const
  *Get the blending mode.*
- bool **GetDepthWrite** () const
  *Get depth write.*
- **Color GetDiffuse** () const
  *Get the diffuse color.*
- **Color GetEmissive** () const
  *Get the emissive color.*
- bool **GetLighting** () const
  *Get lighting enabled.*
- std::string **GetName** () const

*Get the name of the material.*

- double **GetPointSize** () const

  *Get the point size.*

- **ShadeMode GetShadeMode** () const

  *Get the shading mode.*

- double **GetShininess** () const

  *Get the shininess.*

- **Color GetSpecular** () const

  *Get the specular color.*

- std::string **GetTextureImage** () const

  *Get a texture image.*

- double **GetTransparency** () const

  *Get the transparency percentage (0..1)*

- void **SetAmbient** (const **Color** &_clr)

  *Set the ambient color.*

- void **SetBlendFactors** (double _srcFactor, double _dstFactor)

  *Set the blende factors.*

- void **SetBlendMode** (**BlendMode** _b)

  *Set the blending mode.*

- void **SetDepthWrite** (bool _value)

  *Set depth write.*

- void **SetDiffuse** (const **Color** &_clr)

  *Set the diffuse color.*

- void **SetEmissive** (const **Color** &_clr)

  *Set the emissive color.*

- void **SetLighting** (bool _value)

  *Set lighting enabled.*

- void **SetPointSize** (double _size)

  *Set the point size.*

- void **SetShadeMode** (**ShadeMode** _b)

  *Set the shading mode param[in] the shading mode.*

- void **SetShininess** (double _t)

  *Set the shininess.*

- void **SetSpecular** (const **Color** &_clr)

  *Set the specular color.*

- void **SetTextureImage** (const std::string &_tex)

  *Set a texture image.*

- void **SetTextureImage** (const std::string &_tex, const std::string &_resourcePath)

  *Set a texture image.*

- void **SetTransparency** (double _t)

  *Set the transparency percentage (0..1)*

## Static Public Attributes

- static std::string **BlendModeStr** [**BLEND_COUNT**]
- static std::string **ShadeModeStr** [**SHADE_COUNT**]

### Protected Attributes

- **Color ambient**

  *the ambient light color*
- **BlendMode blendMode**

  *blend mode*
- **Color diffuse**

  *the diffuse ligth color*
- **Color emissive**

  *the emissive light color*
- std::string **name**

  *the name of the material*
- double **pointSize**

  *point size*
- **ShadeMode shadeMode**

  *the shade mode*
- double **shininess**

  *shininess value (0 to 1)*
- **Color specular**

  *the specular light color*
- std::string **texImage**

  *the texture image file name*
- double **transparency**

  *transparency value in the range 0 to 1*

### Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Material** &_m)

  *Stream insertion operator param[in] _out the output stream to extract from param[out] _m the material information.*

## 10.103.1 Detailed Description

Encapsulates description of a material.

## 10.103.2 Member Enumeration Documentation

### 10.103.2.1 enum gazebo::common::Material::BlendMode

**Enumerator**

***ADD***

***MODULATE***

***REPLACE***

***BLEND_COUNT***

#### 10.103.2.2 enum gazebo::common::Material::ShadeMode

**Enumerator**

***FLAT***

***GOURAUD***

***PHONG***

***BLINN***

***SHADE_COUNT***

### 10.103.3 Constructor & Destructor Documentation

#### 10.103.3.1 gazebo::common::Material::Material ( )

Constructor.

#### 10.103.3.2 virtual gazebo::common::Material::~Material ( ) [virtual]

Destructor.

#### 10.103.3.3 gazebo::common::Material::Material ( const Color & *_clr* )

Create a material with a default color.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | **Color** (p. 233) of the material |

### 10.103.4 Member Function Documentation

#### 10.103.4.1 Color gazebo::common::Material::GetAmbient ( ) const

Get the ambient color.

**Returns**

The ambient color

#### 10.103.4.2 void gazebo::common::Material::GetBlendFactors ( double & *_srcFactor*, double & *_dstFactor* )

Get the blend factors.

**Parameters**

| | | |
|---|---|---|
| in | *_srcFactor* | Source factor is returned in this variable |
| in | *_dstFactor* | Destination factor is returned in this variable |

#### 10.103.4.3 BlendMode gazebo::common::Material::GetBlendMode ( ) const

Get the blending mode.

**Returns**

the blend mode

#### 10.103.4.4 bool gazebo::common::Material::GetDepthWrite ( ) const

Get depth write.

**Returns**

the depth write enabled state

#### 10.103.4.5 Color gazebo::common::Material::GetDiffuse ( ) const

Get the diffuse color.

**Returns**

The diffuse color

#### 10.103.4.6 Color gazebo::common::Material::GetEmissive ( ) const

Get the emissive color.

**Returns**

The emissive color

#### 10.103.4.7 bool gazebo::common::Material::GetLighting ( ) const

Get lighting enabled.

**Returns**

the lighting enabled state

#### 10.103.4.8 std::string gazebo::common::Material::GetName ( ) const

Get the name of the material.

**Returns**

The name of the material

#### 10.103.4.9 double gazebo::common::Material::GetPointSize ( ) const

Get the point size.

**Returns**

the point size

#### 10.103.4.10 ShadeMode gazebo::common::Material::GetShadeMode ( ) const

Get the shading mode.

**Returns**

the shading mode

#### 10.103.4.11 double gazebo::common::Material::GetShininess ( ) const

Get the shininess.

**Returns**

The shininess value

#### 10.103.4.12 Color gazebo::common::Material::GetSpecular ( ) const

Get the specular color.

**Returns**

The specular color

#### 10.103.4.13 std::string gazebo::common::Material::GetTextureImage ( ) const

Get a texture image.

**Returns**

The name of the texture image (if one exists) or an empty string

#### 10.103.4.14 double gazebo::common::Material::GetTransparency ( ) const

Get the transparency percentage (0..1)

**Returns**

The transparency percentage

#### 10.103.4.15 void gazebo::common::Material::SetAmbient ( const Color & _clr )

Set the ambient color.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | The ambient color |

#### 10.103.4.16 void gazebo::common::Material::SetBlendFactors ( double _srcFactor, double _dstFactor )

Set the blende factors.

Will be interpreted as: (texture ∗ _srcFactor) + (scene_pixel ∗ _dstFactor)

**Parameters**

| | | |
|---|---|---|
| in | *_srcFactor* | The source factor |
| in | *_dstFactor* | The destination factor |

#### 10.103.4.17 void gazebo::common::Material::SetBlendMode ( BlendMode _b )

Set the blending mode.

**Parameters**

| | | |
|---|---|---|
| in | *_b* | the blend mode |

#### 10.103.4.18 void gazebo::common::Material::SetDepthWrite ( bool _value )

Set depth write.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | the depth write enabled state |

#### 10.103.4.19 void gazebo::common::Material::SetDiffuse ( const Color & _clr )

Set the diffuse color.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | The diffuse color |

#### 10.103.4.20 void gazebo::common::Material::SetEmissive ( const Color & _clr )

Set the emissive color.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | The emissive color |

#### 10.103.4.21 void gazebo::common::Material::SetLighting ( bool *_value* )

Set lighting enabled.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | the lighting enabled state |

#### 10.103.4.22 void gazebo::common::Material::SetPointSize ( double *_size* )

Set the point size.

**Parameters**

| | | |
|---|---|---|
| in | *_size* | the size |

#### 10.103.4.23 void gazebo::common::Material::SetShadeMode ( ShadeMode *_b* )

Set the shading mode param[in] the shading mode.

#### 10.103.4.24 void gazebo::common::Material::SetShininess ( double *_t* )

Set the shininess.

**Parameters**

| | | |
|---|---|---|
| in | *_t* | The shininess value |

#### 10.103.4.25 void gazebo::common::Material::SetSpecular ( const Color & *_clr* )

Set the specular color.

**Parameters**

| | | |
|---|---|---|
| in | *_clr* | The specular color |

#### 10.103.4.26 void gazebo::common::Material::SetTextureImage ( const std::string & *_tex* )

Set a texture image.

**Parameters**

| | | |
|---|---|---|
| in | *_tex* | The name of the texture, which must be in Gazebo's resource path |

#### 10.103.4.27 void gazebo::common::Material::SetTextureImage ( const std::string & _tex, const std::string & _resourcePath )

Set a texture image.

**Parameters**

| | | |
|---|---|---|
| in | *_tex* | The name of the texture |
| in | *_resourcePath* | Path which contains _tex |

#### 10.103.4.28 void gazebo::common::Material::SetTransparency ( double _t )

Set the transparency percentage (0..1)

**Parameters**

| | | |
|---|---|---|
| in | *_t* | The amount of transparency (0..1) |

### 10.103.5 Friends And Related Function Documentation

#### 10.103.5.1 std::ostream& operator<< ( std::ostream & _out, const gazebo::common::Material & _m ) [friend]

Stream insertion operator param[in] _out the output stream to extract from param[out] _m the material information.

### 10.103.6 Member Data Documentation

#### 10.103.6.1 Color gazebo::common::Material::ambient [protected]

the ambient light color

#### 10.103.6.2 BlendMode gazebo::common::Material::blendMode [protected]

blend mode

#### 10.103.6.3 std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT] [static]

#### 10.103.6.4 Color gazebo::common::Material::diffuse [protected]

the diffuse ligth color

#### 10.103.6.5 Color gazebo::common::Material::emissive [protected]

the emissive light color

#### 10.103.6.6 std::string gazebo::common::Material::name [protected]

the name of the material

#### 10.103.6.7 double gazebo::common::Material::pointSize [protected]

point size

#### 10.103.6.8 ShadeMode gazebo::common::Material::shadeMode [protected]

the shade mode

#### 10.103.6.9 std::string gazebo::common::Material::ShadeModeStr[SHADE_COUNT] [static]

#### 10.103.6.10 double gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

#### 10.103.6.11 Color gazebo::common::Material::specular [protected]

the specular light color

#### 10.103.6.12 std::string gazebo::common::Material::texImage [protected]

the texture image file name

#### 10.103.6.13 double gazebo::common::Material::transparency [protected]

transparency value in the range 0 to 1

The documentation for this class was generated from the following file:

- **common/Material.hh**

## 10.104 gazebo::math::Matrix3 Class Reference

A 3x3 matrix class.

`#include <Matrix3.hh>`

### Public Member Functions

- **Matrix3** ()

  *Constructor.*
- **Matrix3** (const **Matrix3** &_m)

  *Copy constructor.*
- **Matrix3** (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)

  *Constructor.*
- virtual ~**Matrix3** ()

  *Desctructor.*

- **Matrix3 operator**∗ (const double &_s) const

  *returns the element wise scalar multiplication*
- **Matrix3 operator**∗ (const **Matrix3** &_m) const

  *Matrix multiplication operator.*
- **Matrix3 operator+** (const **Matrix3** &_m) const

  *returns the element wise sum of two matrices*
- **Matrix3 operator-** (const **Matrix3** &_m) const

  *returns the element wise difference of two matrices*
- bool **operator==** (const **Matrix3** &_m) const

  *Equality test operator.*
- const double ∗ **operator[]** (size_t _row) const

  *Array subscript operator.*
- double ∗ **operator[]** (size_t _row)

  *Array subscript operator.*
- void **SetCol** (unsigned int _c, const **Vector3** &_v)

  *Set a column.*
- void **SetFromAxes** (const **Vector3** &_xAxis, const **Vector3** &_yAxis, const **Vector3** &_zAxis)

  *Set the matrix from three axis (1 per column)*
- void **SetFromAxis** (const **Vector3** &_axis, double _angle)

  *Set the matrix from an axis and angle.*

## Protected Attributes

- double **m** [3][3]

  *the 3x3 matrix*

## Friends

- **Matrix3 operator**∗ (double _s, const **Matrix3** &_m)

  *Multiplication operators.*
- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Matrix3** &_m)

  *Stream insertion operator.*

### 10.104.1 Detailed Description

A 3x3 matrix class.

### 10.104.2 Constructor & Destructor Documentation

#### 10.104.2.1 gazebo::math::Matrix3::Matrix3 ( )

Constructor.

Referenced by operator∗(), operator+(), and operator-().

#### 10.104.2.2 gazebo::math::Matrix3::Matrix3 ( const Matrix3 & _m )

Copy constructor.

**Parameters**

| | |
|---|---|
| _m | Matrix to copy |

#### 10.104.2.3 gazebo::math::Matrix3::Matrix3 ( double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22 )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | _v00 | Row 0, Col 0 value |
| in | _v01 | Row 0, Col 1 value |
| in | _v02 | Row 0, Col 2 value |
| in | _v10 | Row 1, Col 0 value |
| in | _v11 | Row 1, Col 1 value |
| in | _v12 | Row 1, Col 2 value |
| in | _v20 | Row 2, Col 0 value |
| in | _v21 | Row 2, Col 1 value |
| in | _v22 | Row 2, Col 2 value |

#### 10.104.2.4 virtual gazebo::math::Matrix3::~Matrix3 ( ) [virtual]

Desctructor.

### 10.104.3 Member Function Documentation

#### 10.104.3.1 Matrix3 gazebo::math::Matrix3::operator* ( const double & _s ) const [inline]

returns the element wise scalar multiplication

References m, and Matrix3().

#### 10.104.3.2 Matrix3 gazebo::math::Matrix3::operator* ( const Matrix3 & _m ) const [inline]

Matrix multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | _m | **Matrix3** (p. 594) to multiply |

**Returns**

product of this * _m

References m, and Matrix3().

#### 10.104.3.3 Matrix3 gazebo::math::Matrix3::operator+ ( const Matrix3 & _m ) const [inline]

returns the element wise sum of two matrices

References m, and Matrix3().

#### 10.104.3.4 Matrix3 gazebo::math::Matrix3::operator- ( const Matrix3 & _m ) const [inline]

returns the element wise difference of two matrices

References m, and Matrix3().

#### 10.104.3.5 bool gazebo::math::Matrix3::operator== ( const Matrix3 & _m ) const

Equality test operator.

**Parameters**

| | | |
|---|---|---|
| in | *_m* | **Matrix3** (p. 594) to test |

**Returns**

True if equal (using the default tolerance of 1e-6)

#### 10.104.3.6 const double* gazebo::math::Matrix3::operator[] ( size_t _row ) const [inline]

Array subscript operator.

**Parameters**

| | | |
|---|---|---|
| in | *_row* | row index |

**Returns**

a pointer to the row

References m.

#### 10.104.3.7 double* gazebo::math::Matrix3::operator[] ( size_t _row ) [inline]

Array subscript operator.

**Parameters**

| | | |
|---|---|---|
| in | *_row* | row index |

**Returns**

a pointer to the row

References m.

#### 10.104.3.8 void gazebo::math::Matrix3::SetCol ( unsigned int _c, const Vector3 & _v )

Set a column.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The colum index (0, 1, 2) |
| in | *_v* | The value to set in each row of the column |

#### 10.104.3.9 void gazebo::math::Matrix3::SetFromAxes ( const Vector3 & _xAxis, const Vector3 & _yAxis, const Vector3 & _zAxis )

Set the matrix from three axis (1 per column)

**Parameters**

| | | |
|---|---|---|
| in | *_xAxis* | The x axis |
| in | *_yAxis* | The y axis |
| in | *_zAxis* | The z axis |

#### 10.104.3.10 void gazebo::math::Matrix3::SetFromAxis ( const Vector3 & _axis, double _angle )

Set the matrix from an axis and angle.

**Parameters**

| | | |
|---|---|---|
| in | *_axis* | the axis |
| in | *_angle* | ccw rotation around the axis in radians |

### 10.104.4 Friends And Related Function Documentation

#### 10.104.4.1 Matrix3 operator* ( double _s, const Matrix3 & _m ) [friend]

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | *_s* | the scaling factor |
| in | *_m* | input matrix |

**Returns**

a scaled matrix

#### 10.104.4.2 std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Matrix3 & _m ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | Output stream |
| in | *_m* | Matrix to output |

**Returns**

the stream

### 10.104.5 Member Data Documentation

#### 10.104.5.1 double gazebo::math::Matrix3::m[3][3] [protected]

the 3x3 matrix

Referenced by operator∗(), operator+(), operator-(), and operator[]().

The documentation for this class was generated from the following file:

- **Matrix3.hh**

## 10.105 gazebo::math::Matrix4 Class Reference

A 3x3 matrix class.

`#include <math/gzmath.hh>`

### Public Member Functions

- **Matrix4** ()

  *Constructor.*
- **Matrix4** (const **Matrix4** &_m)

  *Copy constructor.*
- **Matrix4** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

  *Constructor.*
- virtual ∼**Matrix4** ()

  *Destructor.*
- **math::Pose GetAsPose** () const

  *Get the transformation as **math::Pose** (p. 734).*
- **Vector3 GetEulerRotation** (unsigned int solution_number=1) const

  *Get the rotation as a Euler angles.*
- **Quaternion GetRotation** () const

  *Get the rotation as a quaternion.*
- **Vector3 GetTranslation** () const

  *Get the translational values as a **Vector3** (p. 1091).*
- **Matrix4 Inverse** () const

  *Return the inverse matrix.*
- bool **IsAffine** () const

*Return true if the matrix is affine.*

- **Matrix4 operator**∗ (const **Matrix4** &_mat) const

  *Multiplication operator.*
- **Matrix4 operator**∗ (const **Matrix3** &_mat) const

  *Multiplication operator.*
- **Vector3 operator**∗ (const **Vector3** &_vec) const

  *Multiplication operator.*
- **Matrix4** & **operator=** (const **Matrix4** &_mat)

  *Equal operator.*
- const **Matrix4** & **operator=** (const **Matrix3** &_mat)

  *Equal operator for 3x3 matrix.*
- bool **operator==** (const **Matrix4** &_m) const

  *Equality operator.*
- double ∗ **operator[]** (size_t _row)

  *Array subscript operator.*
- const double ∗ **operator[]** (size_t _row) const
- void **Set** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

  *Change the values.*
- void **SetScale** (const **Vector3** &_s)

  *Set the scale.*
- void **SetTranslate** (const **Vector3** &_t)

  *Set the translational values [ (0, 3) (1, 3) (2, 3) ].*
- **Vector3 TransformAffine** (const **Vector3** &_v) const

  *Perform an affine transformation.*

### Static Public Attributes

- static const **Matrix4 IDENTITY**

  *Identity matrix.*
- static const **Matrix4 ZERO**

  *Zero matrix.*

### Protected Attributes

- double **m** [4][4]

  *The 4x4 matrix.*

### Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Matrix4** &_m)

  *Stream insertion operator.*

### 10.105.1 Detailed Description

A 3x3 matrix class.

## 10.105.2 Constructor & Destructor Documentation

### 10.105.2.1 gazebo::math::Matrix4::Matrix4 ( )

Constructor.

### 10.105.2.2 gazebo::math::Matrix4::Matrix4 ( const Matrix4 & _m )

Copy constructor.

**Parameters**

| | |
|---|---|
| *_m* | Matrix to copy |

### 10.105.2.3 gazebo::math::Matrix4::Matrix4 ( double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33 )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_v00* | Row 0, Col 0 value |
| in | *_v01* | Row 0, Col 1 value |
| in | *_v02* | Row 0, Col 2 value |
| in | *_v03* | Row 0, Col 3 value |
| in | *_v10* | Row 1, Col 0 value |
| in | *_v11* | Row 1, Col 1 value |
| in | *_v12* | Row 1, Col 2 value |
| in | *_v13* | Row 1, Col 3 value |
| in | *_v20* | Row 2, Col 0 value |
| in | *_v21* | Row 2, Col 1 value |
| in | *_v22* | Row 2, Col 2 value |
| in | *_v23* | Row 2, Col 3 value |
| in | *_v30* | Row 3, Col 0 value |
| in | *_v31* | Row 3, Col 1 value |
| in | *_v32* | Row 3, Col 2 value |
| in | *_v33* | Row 3, Col 3 value |

### 10.105.2.4 virtual gazebo::math::Matrix4::~Matrix4 ( ) [virtual]

Destructor.

## 10.105.3 Member Function Documentation

### 10.105.3.1 math::Pose gazebo::math::Matrix4::GetAsPose ( ) const

Get the transformation as **math::Pose** (p. 734).

**Returns**

the pose

#### 10.105.3.2 Vector3 gazebo::math::Matrix4::GetEulerRotation ( unsigned int *solution_number* = `1` ) const

Get the rotation as a Euler angles.

**Returns**

the rotation

#### 10.105.3.3 Quaternion gazebo::math::Matrix4::GetRotation ( ) const

Get the rotation as a quaternion.

**Returns**

the rotation

#### 10.105.3.4 Vector3 gazebo::math::Matrix4::GetTranslation ( ) const

Get the translational values as a **Vector3** (p. 1091).

**Returns**

x,y,z

#### 10.105.3.5 Matrix4 gazebo::math::Matrix4::Inverse ( ) const

Return the inverse matrix.

#### 10.105.3.6 bool gazebo::math::Matrix4::IsAffine ( ) const

Return true if the matrix is affine.

**Returns**

true if the matrix is affine, false otherwise

#### 10.105.3.7 Matrix4 gazebo::math::Matrix4::operator∗ ( const Matrix4 & *_mat* ) const

Multiplication operator.

**Parameters**

| | |
|---|---|
| *_mat* | Incoming matrix |

**Returns**

This matrix * _mat

#### 10.105.3.8 Matrix4 gazebo::math::Matrix4::operator* ( const Matrix3 & _mat ) const

Multiplication operator.

**Parameters**

| | |
|---|---|
| *_mat* | Incoming matrix |

**Returns**

This matrix * _mat

#### 10.105.3.9 Vector3 gazebo::math::Matrix4::operator* ( const Vector3 & _vec ) const

Multiplication operator.

**Parameters**

| | |
|---|---|
| *_vec* | **Vector3** (p. 1091) |

**Returns**

Resulting vector from multiplication

#### 10.105.3.10 Matrix4& gazebo::math::Matrix4::operator= ( const Matrix4 & _mat )

Equal operator.

this = _mat

**Parameters**

| | |
|---|---|
| *_mat* | Incoming matrix |

**Returns**

itself

#### 10.105.3.11 const Matrix4& gazebo::math::Matrix4::operator= ( const Matrix3 & _mat )

Equal operator for 3x3 matrix.

**Parameters**

| | |
|---|---|
| *_mat* | Incoming matrix |

**Returns**

itself

#### 10.105.3.12 bool gazebo::math::Matrix4::operator== ( const Matrix4 & _m ) const

Equality operator.

**Parameters**

| | | |
|---|---|---|
| in | *_m* | **Matrix3** (p. 594) to test |

**Returns**

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

#### 10.105.3.13 double* gazebo::math::Matrix4::operator[] ( size_t _row ) [inline]

Array subscript operator.

**Parameters**

| | | |
|---|---|---|
| in | *_row* | the row index |

**Returns**

the row

References m.

#### 10.105.3.14 const double* gazebo::math::Matrix4::operator[] ( size_t _row ) const [inline]

**Parameters**

| | | |
|---|---|---|
| in | *_row* | the row index |

**Returns**

the row

References m.

#### 10.105.3.15 void gazebo::math::Matrix4::Set ( double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33 )

Change the values.

**Parameters**

| | | |
|---|---|---|
| in | *_v00* | Row 0, Col 0 value |
| in | *_v01* | Row 0, Col 1 value |
| in | *_v02* | Row 0, Col 2 value |
| in | *_v03* | Row 0, Col 3 value |
| in | *_v10* | Row 1, Col 0 value |
| in | *_v11* | Row 1, Col 1 value |
| in | *_v12* | Row 1, Col 2 value |
| in | *_v13* | Row 1, Col 3 value |
| in | *_v20* | Row 2, Col 0 value |
| in | *_v21* | Row 2, Col 1 value |
| in | *_v22* | Row 2, Col 2 value |
| in | *_v23* | Row 2, Col 3 value |
| in | *_v30* | Row 3, Col 0 value |
| in | *_v31* | Row 3, Col 1 value |
| in | *_v32* | Row 3, Col 2 value |
| in | *_v33* | Row 3, Col 3 value |

#### 10.105.3.16 void gazebo::math::Matrix4::SetScale ( const Vector3 & *_s* )

Set the scale.

**Parameters**

| | | |
|---|---|---|
| in | *_s* | scale |

#### 10.105.3.17 void gazebo::math::Matrix4::SetTranslate ( const Vector3 & *_t* )

Set the translational values [ (0, 3) (1, 3) (2, 3) ].

**Parameters**

| | | |
|---|---|---|
| in | *_t* | Values to set |

#### 10.105.3.18 Vector3 gazebo::math::Matrix4::TransformAffine ( const Vector3 & *_v* ) const

Perform an affine transformation.

**Parameters**

| | |
|---|---|
| *_v* | **Vector3** (p. 1091) value for the transformation |

**Returns**

The result of the transformation

### 10.105.4 Friends And Related Function Documentation

#### 10.105.4.1 std::ostream& operator<< ( std::ostream & *_out,* const gazebo::math::Matrix4 & *_m* ) [friend]

Stream insertion operator.

**Parameters**

| | |
|---|---|
| *_out* | output stream |
| *_m* | Matrix to output |

**Returns**

the stream

### 10.105.5 Member Data Documentation

#### 10.105.5.1 const Matrix4 gazebo::math::Matrix4::IDENTITY [static]

Identity matrix.

#### 10.105.5.2 double gazebo::math::Matrix4::m[4][4] [protected]

The 4x4 matrix.

Referenced by operator[]().

#### 10.105.5.3 const Matrix4 gazebo::math::Matrix4::ZERO [static]

Zero matrix.

The documentation for this class was generated from the following file:

- **Matrix4.hh**

## 10.106 gazebo::common::Mesh Class Reference

A 3D mesh.

`#include <common/common.hh>`

### Public Member Functions

- **Mesh** ()
  *Constructor.*
- virtual ~**Mesh** ()
  *Destructor.*
- int **AddMaterial** (**Material** *_mat)
  *Add a material to the mesh.*
- void **AddSubMesh** (**SubMesh** *_child)
  *Add a submesh mesh.*

- void **Center** (const **math::Vector3** &_center=**math::Vector3::Zero**)

  *Move the center of the mesh to the given coordinate.*
- void **FillArrays** (float **_vertArr, int **_indArr) const

  *Put all the data into flat arrays.*
- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

  *Generate texture coordinates using spherical projection from center.*
- void **GetAABB** (**math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz) const

  *Get AABB coordinate.*
- unsigned int **GetIndexCount** () const

  *Return the number of indices.*
- const **Material** * **GetMaterial** (int _index) const

  *Get a material.*
- unsigned int **GetMaterialCount** () const

  *Get the number of materials.*
- **math::Vector3 GetMax** () const

  *Get the maximun X, Y, Z values.*
- **math::Vector3 GetMin** () const

  *Get the minimum X, Y, Z values.*
- std::string **GetName** () const

  *Get the name of this mesh.*
- unsigned int **GetNormalCount** () const

  *Return the number of normals.*
- std::string **GetPath** () const

  *Get the path which contains the mesh resource.*
- **Skeleton** * **GetSkeleton** () const

  *Get the skeleton to which this mesh is attached.*
- const **SubMesh** * **GetSubMesh** (unsigned int _i) const

  *Get a child mesh.*
- const **SubMesh** * **GetSubMesh** (const std::string &_name) const

  *Get a child mesh by name.*
- unsigned int **GetSubMeshCount** () const

  *Get the number of children.*
- unsigned int **GetTexCoordCount** () const

  *Return the number of texture coordinates.*
- unsigned int **GetVertexCount** () const

  *Return the number of vertices.*
- bool **HasSkeleton** () const

  *Return true if mesh is attached to a skeleton.*
- void **RecalculateNormals** ()

  *Recalculate all the normals of each face defined by three indices.*
- void **Scale** (double _factor)

  *Scale all vertices by _factor.*
- void **SetName** (const std::string &_n)

  *Set the name of this mesh.*
- void **SetPath** (const std::string &_path)

  *Set the path which contains the mesh resource.*
- void **SetScale** (const **math::Vector3** &_factor)

*Scale all vertices by the _factor vector.*

- void **SetSkeleton** (**Skeleton** *_skel)

  *Set the mesh skeleton.*

- void **Translate** (const **math::Vector3** &_vec)

  *Move all vertices in all submeshes by _vec.*

### 10.106.1 Detailed Description

A 3D mesh.

### 10.106.2 Constructor & Destructor Documentation

#### 10.106.2.1 gazebo::common::Mesh::Mesh ( )

Constructor.

#### 10.106.2.2 virtual gazebo::common::Mesh::~Mesh ( ) [virtual]

Destructor.

### 10.106.3 Member Function Documentation

#### 10.106.3.1 int gazebo::common::Mesh::AddMaterial ( **Material** * *_mat* )

Add a material to the mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_mat* | the material |

**Returns**

Index of this material

#### 10.106.3.2 void gazebo::common::Mesh::AddSubMesh ( **SubMesh** * *_child* )

Add a submesh mesh.

The **Mesh** (p. 606) object takes ownership of the submesh.

**Parameters**

| | | |
|---|---|---|
| in | *_child* | the submesh |

#### 10.106.3.3 void gazebo::common::Mesh::Center ( const **math::Vector3** & *_center* = **math::Vector3::Zero** )

Move the center of the mesh to the given coordinate.

This will move all the vertices in all submeshes.

**Parameters**

| | | |
|---|---|---|
| in | *_center* | Location of the mesh center. |

#### 10.106.3.4 void gazebo::common::Mesh::FillArrays ( float ** *_vertArr*, int ** *_indArr* ) const

Put all the data into flat arrays.

**Parameters**

| | | |
|---|---|---|
| out | *_vertArr* | the vertex array |
| out | *_indArr* | the index array |

#### 10.106.3.5 void gazebo::common::Mesh::GenSphericalTexCoord ( const math::Vector3 & *_center* )

Generate texture coordinates using spherical projection from center.

**Parameters**

| | | |
|---|---|---|
| in | *_center* | the center of the projection |

#### 10.106.3.6 void gazebo::common::Mesh::GetAABB ( math::Vector3 & *_center*, math::Vector3 & *_min_xyz*, math::Vector3 & *_max_xyz* ) const

Get AABB coordinate.

**Parameters**

| | | |
|---|---|---|
| out | *_center* | of the bounding box |
| out | *_min_xyz* | bounding box minimum values |
| out | *_max_xyz* | bounding box maximum values |

#### 10.106.3.7 unsigned int gazebo::common::Mesh::GetIndexCount ( ) const

Return the number of indices.

**Returns**

the count

#### 10.106.3.8 const Material* gazebo::common::Mesh::GetMaterial ( int *_index* ) const

Get a material.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | the index |

**Returns**

the material or NULL if the index is out of bounds

#### 10.106.3.9 unsigned int gazebo::common::Mesh::GetMaterialCount ( ) const

Get the number of materials.

**Returns**

the count

#### 10.106.3.10 math::Vector3 gazebo::common::Mesh::GetMax ( ) const

Get the maximun X, Y, Z values.

**Returns**

the upper bounds of the bounding box

#### 10.106.3.11 math::Vector3 gazebo::common::Mesh::GetMin ( ) const

Get the minimum X, Y, Z values.

**Returns**

the lower bounds of the bounding box

#### 10.106.3.12 std::string gazebo::common::Mesh::GetName ( ) const

Get the name of this mesh.

**Returns**

the name

#### 10.106.3.13 unsigned int gazebo::common::Mesh::GetNormalCount ( ) const

Return the number of normals.

**Returns**

the count

#### 10.106.3.14 std::string gazebo::common::Mesh::GetPath ( ) const

Get the path which contains the mesh resource.

**Returns**

the path to the mesh resource

#### 10.106.3.15 Skeleton* gazebo::common::Mesh::GetSkeleton ( ) const

Get the skeleton to which this mesh is attached.

**Returns**

pointer to skeleton, or NULL if none is present.

#### 10.106.3.16 const SubMesh* gazebo::common::Mesh::GetSubMesh ( unsigned int _i ) const

Get a child mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | the index |

**Returns**

the submesh. An exception is thrown if the index is out of bounds

#### 10.106.3.17 const SubMesh* gazebo::common::Mesh::GetSubMesh ( const std::string & _name ) const

Get a child mesh by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the submesh. |

**Returns**

The submesh, NULL if the _name is not found.

#### 10.106.3.18 unsigned int gazebo::common::Mesh::GetSubMeshCount ( ) const

Get the number of children.

**Returns**

the count

#### 10.106.3.19 unsigned int gazebo::common::Mesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

**Returns**

the count

#### 10.106.3.20 unsigned int gazebo::common::Mesh::GetVertexCount ( ) const

Return the number of vertices.

**Returns**

the count

#### 10.106.3.21 bool gazebo::common::Mesh::HasSkeleton ( ) const

Return true if mesh is attached to a skeleton.

#### 10.106.3.22 void gazebo::common::Mesh::RecalculateNormals ( )

Recalculate all the normals of each face defined by three indices.

#### 10.106.3.23 void gazebo::common::Mesh::Scale ( double *_factor* )

Scale all vertices by _factor.

**Parameters**

| | |
|---|---|
| *_factor* | Scaling factor |

#### 10.106.3.24 void gazebo::common::Mesh::SetName ( const std::string & *_n* )

Set the name of this mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_n* | the name to set |

#### 10.106.3.25 void gazebo::common::Mesh::SetPath ( const std::string & *_path* )

Set the path which contains the mesh resource.

**Parameters**

| | | |
|---|---|---|
| in | *_path* | the file path |

#### 10.106.3.26 void gazebo::common::Mesh::SetScale ( const **math::Vector3** & *_factor* )

Scale all vertices by the _factor vector.

**Parameters**

| | | |
|---|---|---|
| in | *_factor* | Scaling vector |

#### 10.106.3.27 void gazebo::common::Mesh::SetSkeleton ( Skeleton * _skel )

Set the mesh skeleton.

#### 10.106.3.28 void gazebo::common::Mesh::Translate ( const math::Vector3 & _vec )

Move all vertices in all submeshes by _vec.

**Parameters**

| | | |
|---|---|---|
| in | *_vec* | Amount to translate vertices. |

The documentation for this class was generated from the following file:

- **Mesh.hh**

## 10.107 gazebo::common::MeshCSG Class Reference

Creates CSG meshes.

`#include <common/common.hh>`

### Public Types

- enum **BooleanOperation** { **UNION**, **INTERSECTION**, **DIFFERENCE** }

  *An enumeration of the boolean operations.*

### Public Member Functions

- **MeshCSG** ()

  *Constructor.*
- virtual ~**MeshCSG** ()

  *Destructor.*
- **Mesh** * **CreateBoolean** (const **Mesh** *_m1, const **Mesh** *_m2, const int _operation, const **math::Pose** &_offset=**math::Pose::Zero**)

  *Create a boolean mesh from two meshes.*

### 10.107.1 Detailed Description

Creates CSG meshes.

### 10.107.2 Member Enumeration Documentation

#### 10.107.2.1 enum gazebo::common::MeshCSG::BooleanOperation

An enumeration of the boolean operations.

**Enumerator**

***UNION***

***INTERSECTION***

***DIFFERENCE***

### 10.107.3 Constructor & Destructor Documentation

#### 10.107.3.1 gazebo::common::MeshCSG::MeshCSG ( )

Constructor.

#### 10.107.3.2 virtual gazebo::common::MeshCSG::~MeshCSG ( ) [virtual]

Destructor.

### 10.107.4 Member Function Documentation

#### 10.107.4.1 Mesh* gazebo::common::MeshCSG::CreateBoolean ( const Mesh * *_m1*, const Mesh * *_m2*, const int *_operation*, const math::Pose & *_offset* = math::Pose::Zero )

Create a boolean mesh from two meshes.

**Parameters**

| | | |
|---|---|---|
| in | *_m1* | the parent mesh in the boolean operation |
| in | *_m2* | the child mesh in the boolean operation |
| in | *_operation* | the boolean operation applied to the two meshes |
| in | *_offset* | _m2's pose offset from _m1 |

**Returns**

a pointer to the created mesh

The documentation for this class was generated from the following file:

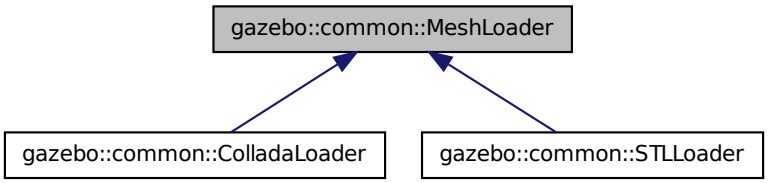
- **MeshCSG.hh**

## 10.108 gazebo::common::MeshLoader Class Reference

Base class for loading meshes.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::MeshLoader:




## Public Member Functions

- **MeshLoader** ()
  *Constructor.*
- virtual ∼**MeshLoader** ()
  *Destructor.*
- virtual **Mesh** ∗ **Load** (const std::string &_filename)=0
  *Load a 3D mesh.*

### 10.108.1 Detailed Description

Base class for loading meshes.

### 10.108.2 Constructor & Destructor Documentation

#### 10.108.2.1 gazebo::common::MeshLoader::MeshLoader ( )

Constructor.

#### 10.108.2.2 virtual gazebo::common::MeshLoader::∼MeshLoader ( ) [virtual]

Destructor.

### 10.108.3 Member Function Documentation

#### 10.108.3.1 virtual Mesh∗ gazebo::common::MeshLoader::Load ( const std::string & *_filename* ) [pure virtual]

Load a 3D mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | the path to the mesh |

**Returns**

a pointer to the created mesh

Implemented in **gazebo::common::ColladaLoader** (p. 219), and **gazebo::common::STLLoader** (p. 1003).

The documentation for this class was generated from the following file:

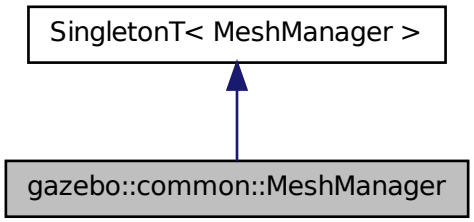
- **MeshLoader.hh**

## 10.109 gazebo::common::MeshManager Class Reference

Maintains and manages all meshes.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::MeshManager:




### Public Member Functions

- void **AddMesh** (**Mesh** *_mesh)

  *Add a mesh to the manager.*
- void **CreateBox** (const std::string &_name, const **math::Vector3** &_sides, const **math::Vector2d** &_uvCoords)

  *Create a Box mesh.*
- void **CreateCamera** (const std::string &_name, float _scale)

  *Create a Camera mesh.*
- void **CreateCone** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

  *Create a cone mesh.*
- void **CreateCylinder** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

  *Create a cylinder mesh.*
- void **CreatePlane** (const std::string &_name, const **math::Plane** &_plane, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

  *Create mesh for a plane.*
- void **CreatePlane** (const std::string &_name, const **math::Vector3** &_normal, double _d, const **math::Vector2d** &_size, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

  *Create mesh for a plane.*

- void **CreateSphere** (const std::string &_name, float _radius, int _rings, int _segments)

  *Create a sphere mesh.*
- void **CreateTube** (const std::string &_name, float _innerRadius, float _outterRadius, float _height, int _rings, int _segments)

  *Create a tube mesh.*
- void **GenSphericalTexCoord** (const **Mesh** *_mesh, **math::Vector3** _center)

  *generate spherical texture coordinates*
- const **Mesh** * **GetMesh** (const std::string &_name) const

  *Get a mesh by name.*
- void **GetMeshAABB** (const **Mesh** *_mesh, **math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz)

  *Get mesh aabb and center.*
- bool **HasMesh** (const std::string &_name) const

  *Return true if the mesh exists.*
- bool **IsValidFilename** (const std::string &_filename)

  *Checks a path extension against the list of valid extensions.*
- const **Mesh** * **Load** (const std::string &_filename)

  *Load a mesh from a file.*

## Additional Inherited Members

### 10.109.1 Detailed Description

Maintains and manages all meshes.

### 10.109.2 Member Function Documentation

#### 10.109.2.1 void gazebo::common::MeshManager::AddMesh ( Mesh * *_mesh* )

Add a mesh to the manager.

This **MeshManager** (p. 616) takes ownership of the mesh and will destroy it. See ~MeshManager.

**Parameters**

| | | |
|---|---|---|
| in | *the* | mesh to add. |

#### 10.109.2.2 void gazebo::common::MeshManager::CreateBox ( const std::string & *_name,* const math::Vector3 & *_sides,* const math::Vector2d & *_uvCoords* )

Create a Box mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the new mesh |
| in | *_sides* | the x y x dimentions of eah side in meter |
| in | *_uvCoords* | the texture coordinates |

#### 10.109.2.3 void gazebo::common::MeshManager::CreateCamera ( const std::string & *_name,* float *_scale* )

Create a Camera mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | name of the new mesh |
| in | *_scale* | scaling factor for the camera |

#### 10.109.2.4 void gazebo::common::MeshManager::CreateCone ( const std::string & *_name,* float *_radius,* float *_height,* int *_rings,* int *_segments* )

Create a cone mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the new mesh |
| in | *_radius* | the radius of the cylinder in the x y plane |
| in | *_height* | the height along z |
| in | *_rings* | the number of circles along the height |
| in | *_segments* | the number of segment per circle |

#### 10.109.2.5 void gazebo::common::MeshManager::CreateCylinder ( const std::string & *_name,* float *_radius,* float *_height,* int *_rings,* int *_segments* )

Create a cylinder mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the new mesh |
| in | *_radius* | the radius of the cylinder in the x y plane |
| in | *_height* | the height along z |
| in | *_rings* | the number of circles along the height |
| in | *_segments* | the number of segment per circle |

#### 10.109.2.6 void gazebo::common::MeshManager::CreatePlane ( const std::string & *_name,* const **math::Plane** & *_plane,* const **math::Vector2d** & *_segments,* const **math::Vector2d** & *_uvTile* )

Create mesh for a plane.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | |
| in | *_plane* | plane parameters |
| in | *_segments* | number of segments in x and y |
| in | *_uvTile* | the texture tile size in x and y |

#### 10.109.2.7 void gazebo::common::MeshManager::CreatePlane ( const std::string & *_name*, const math::Vector3 & *_normal*, double *_d*, const math::Vector2d & *_size*, const math::Vector2d & *_segments*, const math::Vector2d & *_uvTile* )

Create mesh for a plane.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the new mesh |
| in | *_normal* | the normal to the plane |
| in | *_d* | distance from the origin along normal |
| in | *_size* | the size of the plane in x and y |
| in | *_segments* | the number of segments in x and y |
| in | *_uvTile* | the texture tile size in x and y |

#### 10.109.2.8 void gazebo::common::MeshManager::CreateSphere ( const std::string & *_name*, float *_radius*, int *_rings*, int *_segments* )

Create a sphere mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the mesh |
| in | *_radius* | radius of the sphere in meter |
| in | *_rings* | number of circles on th y axis |
| in | *_segments* | number of segment per circle |

#### 10.109.2.9 void gazebo::common::MeshManager::CreateTube ( const std::string & *_name*, float *_innerRadius*, float *_outterRadius*, float *_height*, int *_rings*, int *_segments* )

Create a tube mesh.

Generates rings inside and outside the cylinder Needs at least two rings and 3 segments

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the new mesh |
| in | *_innerRadius* | the inner radius of the tube in the x y plane |
| in | *_outterRadius* | the outer radius of the tube in the x y plane |
| in | *_height* | the height along z |
| in | *_rings* | the number of circles along the height |
| in | *_segments* | the number of segment per circle |

#### 10.109.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord ( const Mesh * *_mesh*, math::Vector3 *_center* )

generate spherical texture coordinates

#### 10.109.2.11 const Mesh* gazebo::common::MeshManager::GetMesh ( const std::string & *_name* ) const

Get a mesh by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the mesh to look for |

**Returns**

the mesh or NULL if not found

#### 10.109.2.12 void gazebo::common::MeshManager::GetMeshAABB ( const Mesh * *_mesh*, math::Vector3 & *_center*, math::Vector3 & *_min_xyz*, math::Vector3 & *_max_xyz* )

Get mesh aabb and center.

**Parameters**

| | | |
|---|---|---|
| in | *_mesh* | the mesh |
| out | *_center* | the AAB center position |
| out | *_min_xyz* | the bounding box minimum |
| out | *_max_xyz* | the bounding box maximum |

#### 10.109.2.13 bool gazebo::common::MeshManager::HasMesh ( const std::string & *_name* ) const

Return true if the mesh exists.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the mesh |

#### 10.109.2.14 bool gazebo::common::MeshManager::IsValidFilename ( const std::string & *_filename* )

Checks a path extension against the list of valid extensions.

**Returns**

true if the file extension is loadable

#### 10.109.2.15 const Mesh* gazebo::common::MeshManager::Load ( const std::string & *_filename* )

Load a mesh from a file.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | the path to the mesh |

**Returns**

a pointer to the created mesh

The documentation for this class was generated from the following file:

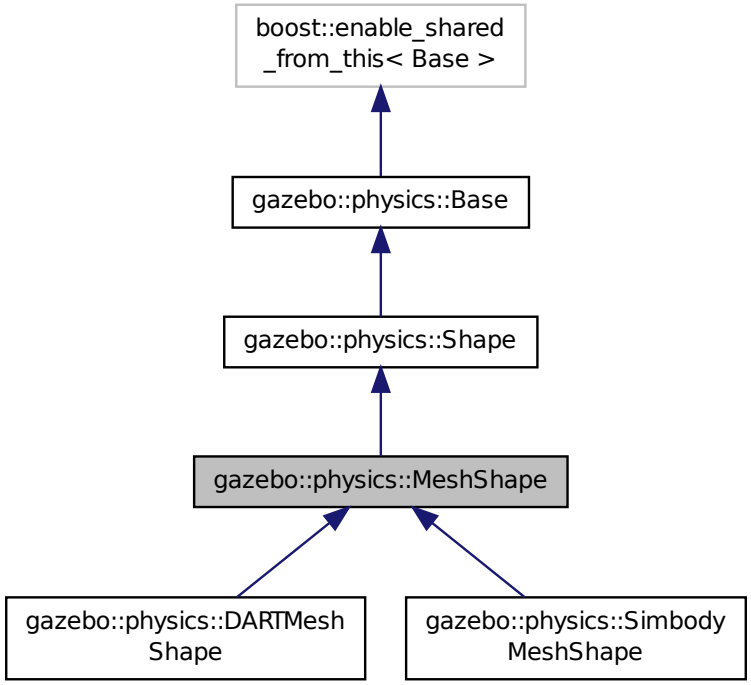
- **MeshManager.hh**

## 10.110 gazebo::physics::MeshShape Class Reference

Triangle mesh collision shape.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::MeshShape:




### Public Member Functions

- **MeshShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**MeshShape** ()

  *Destructor.*
- void **FillMsg** (msgs::Geometry &_msg)

  *Populate a msgs::Geometry message with data from this shape.*
- std::string **GetMeshURI** () const

  *Get the URI of the mesh data.*
- virtual **math::Vector3 GetSize** () const

  *Get the size of the triangle mesh.*
- virtual void **Init** ()

  *Initialize the shape.*
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

*Update this shape from a message.*

- void **SetMesh** (const std::string &_uri, const std::string &_submesh="", bool _center=false)

  *Set the mesh uri and submesh name.*
- void **SetScale** (const **math::Vector3** &_scale)

  *Set the scaling factor.*
- virtual void **Update** ()

  *Update the tri mesh.*

### Protected Attributes

- const **common::Mesh** * **mesh**

  *Pointer to the mesh data.*
- **common::SubMesh** * **submesh**

  *The submesh to use from within the parent mesh.*

### Additional Inherited Members

### 10.110.1 Detailed Description

Triangle mesh collision shape.

### 10.110.2 Constructor & Destructor Documentation

#### 10.110.2.1 gazebo::physics::MeshShape::MeshShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent collision. |

#### 10.110.2.2 virtual gazebo::physics::MeshShape::~MeshShape ( ) [virtual]

Destructor.

### 10.110.3 Member Function Documentation

#### 10.110.3.1 void gazebo::physics::MeshShape::FillMsg ( msgs::Geometry & *_msg* ) [virtual]

Populate a msgs::Geometry message with data from this shape.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message to fill. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.110.3.2 std::string gazebo::physics::MeshShape::GetMeshURI ( ) const

Get the URI of the mesh data.

**Returns**

The URI of the mesh data.

#### 10.110.3.3 virtual math::Vector3 gazebo::physics::MeshShape::GetSize ( ) const [virtual]

Get the size of the triangle mesh.

**Returns**

The size of the triangle mesh.

#### 10.110.3.4 virtual void gazebo::physics::MeshShape::Init ( ) [virtual]

Initialize the shape.

Implements **gazebo::physics::Shape** (p. 864).

Reimplemented in **gazebo::physics::SimbodyMeshShape** (p. 911), and **gazebo::physics::DARTMeshShape** (p. 325).

#### 10.110.3.5 virtual void gazebo::physics::MeshShape::ProcessMsg ( const msgs::Geometry & *_msg* ) [virtual]

Update this shape from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message that contains triangle mesh info. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.110.3.6 void gazebo::physics::MeshShape::SetMesh ( const std::string & *_uri*, const std::string & *_submesh* = " ", bool *_center* = false )

Set the mesh uri and submesh name.

**Parameters**

| | | |
|---|---|---|
| in | *_uri* | Filename of the mesh file to load from. |
| in | *_submesh* | Name of the submesh to use within the mesh |
| in | *_center* | True to center the submesh. specified in the _uri. |

#### 10.110.3.7 void gazebo::physics::MeshShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set the scaling factor.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scaling factor. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.110.3.8 virtual void gazebo::physics::MeshShape::Update ( ) [inline], [virtual]

Update the tri mesh.

Reimplemented from **gazebo::physics::Base** (p. 171).

Reimplemented in **gazebo::physics::DARTMeshShape** (p. 326).

### 10.110.4 Member Data Documentation

#### 10.110.4.1 const common::Mesh* gazebo::physics::MeshShape::mesh [protected]

Pointer to the mesh data.

#### 10.110.4.2 common::SubMesh* gazebo::physics::MeshShape::submesh [protected]

The submesh to use from within the parent mesh.

The documentation for this class was generated from the following file:

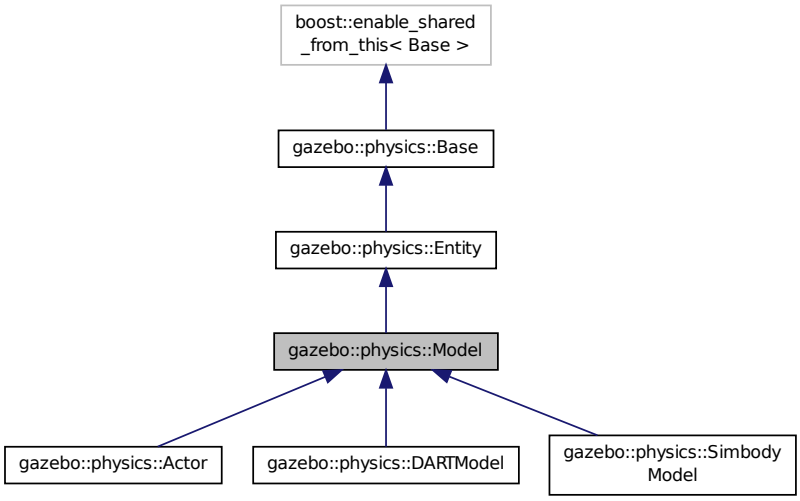
- **MeshShape.hh**

## 10.111 gazebo::physics::Model Class Reference

A model is a collection of links, joints, and plugins.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Model:




## Public Member Functions

- **Model** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**Model** ()

  *Destructor.*
- void **AttachStaticModel** (**ModelPtr** &_model, **math::Pose** _offset)

  *Attach a static model to this model.*
- void **DetachStaticModel** (const std::string &_model)

  *Detach a static model from this model.*
- virtual void **FillMsg** (msgs::Model &_msg)

  *Fill a model message.*
- virtual void **Fini** ()

  *Finalize the model.*
- bool **GetAutoDisable** () const

  *Return the value of the SDF <allow_auto_disable> element.*
- virtual **math::Box GetBoundingBox** () const

  *Get the size of the bounding box.*
- **GripperPtr GetGripper** (size_t _index) const

  *Get a gripper based on an index.*
- size_t **GetGripperCount** () const

  *Get the number of grippers in this model.*
- **JointPtr GetJoint** (const std::string &name)

  *Get a joint.*
- **JointControllerPtr GetJointController** ()

  *Get a handle to the Controller for the joints in this model.*

- unsigned int **GetJointCount** () const

  *Get the number of joints.*
- const **Joint_V** & **GetJoints** () const

  *Get the joints.*
- **LinkPtr GetLink** (const std::string &_name="canonical") const

  *Get a link by name.*
- **Link_V GetLinks** () const

  *Construct and return a vector of **Link** (p. 542)'s in this model Note this constructs the vector of **Link** (p. 542)'s on the fly, could be costly.*
- unsigned int **GetPluginCount** () const

  *Get the number of plugins this model has.*
- virtual **math::Vector3 GetRelativeAngularAccel** () const

  *Get the angular acceleration of the entity.*
- virtual **math::Vector3 GetRelativeAngularVel** () const

  *Get the angular velocity of the entity.*
- virtual **math::Vector3 GetRelativeLinearAccel** () const

  *Get the linear acceleration of the entity.*
- virtual **math::Vector3 GetRelativeLinearVel** () const

  *Get the linear velocity of the entity.*
- virtual const sdf::ElementPtr **GetSDF** ()

  *Get the SDF values for the model.*
- unsigned int **GetSensorCount** () const

  *Get the number of sensors attached to this model.*
- virtual **math::Vector3 GetWorldAngularAccel** () const

  *Get the angular acceleration of the entity in the world frame.*
- virtual **math::Vector3 GetWorldAngularVel** () const

  *Get the angular velocity of the entity in the world frame.*
- virtual **math::Vector3 GetWorldLinearAccel** () const

  *Get the linear acceleration of the entity in the world frame.*
- virtual **math::Vector3 GetWorldLinearVel** () const

  *Get the linear velocity of the entity in the world frame.*
- virtual void **Init** ()

  *Initialize the model.*
- void **Load** (sdf::ElementPtr _sdf)

  *Load the model.*
- void **LoadJoints** ()

  *Load all the joints.*
- void **LoadPlugins** ()

  *Load all plugins.*
- void **ProcessMsg** (const msgs::Model &_msg)

  *Update parameters from a model message.*
- virtual void **RemoveChild** (**EntityPtr** _child)

  *Remove a child.*
- void **Reset** ()

  *Reset the model.*
- void **SetAngularAccel** (const **math::Vector3** &_vel)

  *Set the angular acceleration of the model, and all its links.*

- void **SetAngularVel** (const **math::Vector3** &_vel)

  *Set the angular velocity of the model, and all its links.*
- void **SetAutoDisable** (bool _disable)

  *Allow the model the auto disable.*
- void **SetCollideMode** (const std::string &_mode)

  *This is not implemented in **Link** (p. 542), which means this function doesn't do anything.*
- void **SetEnabled** (bool _enabled)

  *Enable all the links in all the models.*
- void **SetGravityMode** (const bool &_value)

  *Set the gravity mode of the model.*
- void **SetJointAnimation** (const std::map< std::string, **common::NumericAnimationPtr** > _anim, boost::function< void()> _onComplete=**NULL**)

  ***Joint** (p. 496) Animation.*
- void **SetJointPosition** (const std::string &_jointName, double _position, int _index=0)

  *Set the positions of a **Joint** (p. 496) by name.*
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

  *Set the positions of a set of joints.*
- void **SetLaserRetro** (const float _retro)

  *Set the laser retro reflectiveness of the model.*
- void **SetLinearAccel** (const **math::Vector3** &_vel)

  *Set the linear acceleration of the model, and all its links.*
- void **SetLinearVel** (const **math::Vector3** &_vel)

  *Set the linear velocity of the model, and all its links.*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, std::string _linkName)

  *Set the Pose of the entire **Model** (p. 624) by specifying desired Pose of a **Link** (p. 542) within the **Model** (p. 624).*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)

  *Set the Pose of the entire **Model** (p. 624) by specifying desired Pose of a **Link** (p. 542) within the **Model** (p. 624).*
- void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of model.*
- void **SetState** (const **ModelState** &_state)

  *Set the current model state.*
- virtual void **StopAnimation** ()

  *Stop the current animations.*
- void **Update** ()

  *Update the model.*
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  *Update the parameters using new sdf values.*

## Protected Member Functions

- virtual void **OnPoseChange** ()

  *Callback when the pose of the model has been changed.*

### Protected Attributes

- std::vector< **ModelPtr** > **attachedModels**

  *used by **Model::AttachStaticModel** (p. 628)*
- std::vector< **math::Pose** > **attachedModelsOffset**

  *used by **Model::AttachStaticModel** (p. 628)*
- **transport::PublisherPtr jointPub**

  *Publisher for joint info.*

### Additional Inherited Members

## 10.111.1 Detailed Description

A model is a collection of links, joints, and plugins.

## 10.111.2 Constructor & Destructor Documentation

#### 10.111.2.1 gazebo::physics::Model::Model ( **BasePtr** *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent object. |

#### 10.111.2.2 virtual gazebo::physics::Model::~Model ( ) [virtual]

Destructor.

## 10.111.3 Member Function Documentation

#### 10.111.3.1 void gazebo::physics::Model::AttachStaticModel ( **ModelPtr** & *_model,* **math::Pose** *_offset* )

Attach a static model to this model.

This function takes as input a static **Model** (p. 624), which is a **Model** (p. 624) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 624) with a given offset.

This function is useful when you want to simulate a grasp of a static object, or move a static object around using a dynamic model.

If you are in doubt, do not use this function.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the static model. |
| in | *_offset* | Offset, relative to this **Model** (p. 624), to place _model. |

#### 10.111.3.2 void gazebo::physics::Model::DetachStaticModel ( const std::string & *_model* )

Detach a static model from this model.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Name of an attached static model to remove. |

**See Also**

**Model::AttachStaticModel** (p. 628).

#### 10.111.3.3 virtual void gazebo::physics::Model::FillMsg ( msgs::Model & *_msg* ) [virtual]

Fill a model message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to fill using this model's data. |

#### 10.111.3.4 virtual void gazebo::physics::Model::Fini ( ) [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 382).

Reimplemented in **gazebo::physics::Actor** (p. 134), and **gazebo::physics::DARTModel** (p. 328).

#### 10.111.3.5 bool gazebo::physics::Model::GetAutoDisable ( ) const

Return the value of the SDF <allow_auto_disable> element.

**Returns**

True if auto disable is allowed for this model.

#### 10.111.3.6 virtual math::Box gazebo::physics::Model::GetBoundingBox ( ) const [virtual]

Get the size of the bounding box.

**Returns**

The bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 382).

#### 10.111.3.7 GripperPtr gazebo::physics::Model::GetGripper ( size_t *_index* ) const

Get a gripper based on an index.

**Returns**

A pointer to a **Gripper** (p. 453). Null if the _index is invalid.

#### 10.111.3.8 size_t gazebo::physics::Model::GetGripperCount ( ) const

Get the number of grippers in this model.

**Returns**

Size of this->grippers array.

**See Also**

**Model::GetGripper()** (p. 629)

#### 10.111.3.9 JointPtr gazebo::physics::Model::GetJoint ( const std::string & *name* )

Get a joint.

**Parameters**

| | |
|---|---|
| *name* | The name of the joint, specified in the world file |

**Returns**

Pointer to the joint

#### 10.111.3.10 JointControllerPtr gazebo::physics::Model::GetJointController ( )

Get a handle to the Controller for the joints in this model.

**Returns**

A handle to the Controller for the joints in this model.

#### 10.111.3.11 unsigned int gazebo::physics::Model::GetJointCount ( ) const

Get the number of joints.

**Returns**

Get the number of joints.

#### 10.111.3.12 const Joint_V& gazebo::physics::Model::GetJoints ( ) const

Get the joints.

**Returns**

Vector of joints.

#### 10.111.3.13 LinkPtr gazebo::physics::Model::GetLink ( const std::string & _name = "canonical" ) const

Get a link by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the link to get. |

**Returns**

Pointer to the link, NULL if the name is invalid.

#### 10.111.3.14 Link_V gazebo::physics::Model::GetLinks ( ) const

Construct and return a vector of **Link** (p. 542)'s in this model Note this constructs the vector of **Link** (p. 542)'s on the fly, could be costly.

**Returns**

a vector of **Link** (p. 542)'s in this model

#### 10.111.3.15 unsigned int gazebo::physics::Model::GetPluginCount ( ) const

Get the number of plugins this model has.

**Returns**

Number of plugins associated with this model.

#### 10.111.3.16 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.111.3.17 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the entity.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.111.3.18 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.111.3.19 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the entity.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 384).

#### 10.111.3.20 virtual const sdf::ElementPtr gazebo::physics::Model::GetSDF ( ) [virtual]

Get the SDF values for the model.

**Returns**

The SDF value for this model.

Reimplemented from **gazebo::physics::Base** (p. 167).

Reimplemented in **gazebo::physics::Actor** (p. 134).

#### 10.111.3.21 unsigned int gazebo::physics::Model::GetSensorCount ( ) const

Get the number of sensors attached to this model.

This will count all the sensors attached to all the links.

**Returns**

Number of sensors.

#### 10.111.3.22 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity in the world frame.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.111.3.23 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.111.3.24 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity in the world frame.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.111.3.25 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the entity in the world frame.

**Returns**

**math::Vector3** (p. 1091), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.111.3.26 virtual void gazebo::physics::Model::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 167).

Reimplemented in **gazebo::physics::Actor** (p. 134), **gazebo::physics::DARTModel** (p. 328), and **gazebo::physics::SimbodyModel** (p. 913).

#### 10.111.3.27 void gazebo::physics::Model::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the model.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters to load from. |

Reimplemented from **gazebo::physics::Entity** (p. 386).

Reimplemented in **gazebo::physics::SimbodyModel** (p. 913).

#### 10.111.3.28 void gazebo::physics::Model::LoadJoints ( )

Load all the joints.

#### 10.111.3.29 void gazebo::physics::Model::LoadPlugins ( )

Load all plugins.

Load all plugins specified in the SDF for the model.

#### 10.111.3.30 virtual void gazebo::physics::Model::OnPoseChange ( ) [protected], [virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 386).

#### 10.111.3.31 void gazebo::physics::Model::ProcessMsg ( const msgs::Model & *_msg* )

Update parameters from a model message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to process. |

#### 10.111.3.32 virtual void gazebo::physics::Model::RemoveChild ( EntityPtr *_child* ) [virtual]

Remove a child.

**Parameters**

| | | |
|---|---|---|
| in | *_child* | Remove a child entity. |

#### 10.111.3.33 void gazebo::physics::Model::Reset ( ) [virtual]

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 387).

#### 10.111.3.34 void gazebo::physics::Model::SetAngularAccel ( const math::Vector3 & *_vel* )

Set the angular acceleration of the model, and all its links.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | The new angular acceleration |

#### 10.111.3.35 void gazebo::physics::Model::SetAngularVel ( const math::Vector3 & *_vel* )

Set the angular velocity of the model, and all its links.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | The new angular velocity. |

#### 10.111.3.36 void gazebo::physics::Model::SetAutoDisable ( bool *_disable* )

Allow the model the auto disable.

This is ignored if the model has joints.

**Parameters**

| | | |
|---|---|---|
| in | *_disable* | If true, the model is allowed to auto disable. |

#### 10.111.3.37 void gazebo::physics::Model::SetCollideMode ( const std::string & *_mode* )

This is not implemented in **Link** (p. 542), which means this function doesn't do anything.

Set the collide mode of the model.

**Parameters**

| | | |
|---|---|---|
| in | *_mode* | The collision mode |

#### 10.111.3.38 void gazebo::physics::Model::SetEnabled ( bool *_enabled* )

Enable all the links in all the models.

**Parameters**

| | | |
|---|---|---|
| in | *_enabled* | True to enable all the links. |

#### 10.111.3.39 void gazebo::physics::Model::SetGravityMode ( const bool & *_value* )

Set the gravity mode of the model.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | False to turn gravity on for the model. |

#### 10.111.3.40 void gazebo::physics::Model::SetJointAnimation ( const std::map< std::string, common::NumericAnimationPtr > *_anim*, boost::function< void() > *_onComplete* = NULL )

**Joint** (p. 496) Animation.

**Parameters**

| | | |
|---|---|---|
| in | *_anim* | Map of joint names to their position animation. |
| in | *_onComplete* | Callback function for when the animation completes. |

#### 10.111.3.41 void gazebo::physics::Model::SetJointPosition ( const std::string & _jointName, double _position, int _index = 0 )

Set the positions of a **Joint** (p. 496) by name.

**See Also**

**JointController::SetJointPosition** (p. 522)

**Parameters**

| | | |
|---|---|---|
| in | *_jointName* | Name of the joint to set. |
| in | *_position* | Position to set the joint to. |

#### 10.111.3.42 void gazebo::physics::Model::SetJointPositions ( const std::map< std::string, double > & _jointPositions )

Set the positions of a set of joints.

**See Also**

**JointController::SetJointPositions** (p. 522).

**Parameters**

| | | |
|---|---|---|
| in | *_jointPositions* | Map of joint names to their positions. |

#### 10.111.3.43 void gazebo::physics::Model::SetLaserRetro ( const float _retro )

Set the laser retro reflectiveness of the model.

**Parameters**

| | | |
|---|---|---|
| in | *_retro* | Retro reflectance value. |

#### 10.111.3.44 void gazebo::physics::Model::SetLinearAccel ( const math::Vector3 & _vel )

Set the linear acceleration of the model, and all its links.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | The new linear acceleration. |

#### 10.111.3.45 void gazebo::physics::Model::SetLinearVel ( const math::Vector3 & _vel )

Set the linear velocity of the model, and all its links.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | The new linear velocity. |

#### 10.111.3.46 void gazebo::physics::Model::SetLinkWorldPose ( const math::Pose & _pose, std::string _linkName )

Set the Pose of the entire **Model** (p. 624) by specifying desired Pose of a **Link** (p. 542) within the **Model** (p. 624).

Doing so, keeps the configuration of the **Model** (p. 624) unchanged, i.e. all **Joint** (p. 496) angles are unchanged.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | Pose to set the link to. |
| in | *_linkName* | Name of the link to set. |

#### 10.111.3.47 void gazebo::physics::Model::SetLinkWorldPose ( const math::Pose & _pose, const LinkPtr & _link )

Set the Pose of the entire **Model** (p. 624) by specifying desired Pose of a **Link** (p. 542) within the **Model** (p. 624).

Doing so, keeps the configuration of the **Model** (p. 624) unchanged, i.e. all **Joint** (p. 496) angles are unchanged.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | Pose to set the link to. |
| in | *_link* | Pointer to the link to set. |

#### 10.111.3.48 void gazebo::physics::Model::SetScale ( const math::Vector3 & _scale )

Set the scale of model.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the model to. |

#### 10.111.3.49 void gazebo::physics::Model::SetState ( const ModelState & _state )

Set the current model state.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **State** (p. 998) to set the model to. |

#### 10.111.3.50 virtual void gazebo::physics::Model::StopAnimation ( ) [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 389).

#### 10.111.3.51 void gazebo::physics::Model::Update ( ) [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 171).

#### 10.111.3.52 virtual void gazebo::physics::Model::UpdateParameters ( sdf::ElementPtr _sdf ) [virtual]

Update the parameters using new sdf values.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to update from. |

Reimplemented from **gazebo::physics::Entity** (p. 389).

Reimplemented in **gazebo::physics::Actor** (p. 135).

### 10.111.4 Member Data Documentation

#### 10.111.4.1 std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]

used by **Model::AttachStaticModel** (p. 628)

#### 10.111.4.2 std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset [protected]

used by **Model::AttachStaticModel** (p. 628)

#### 10.111.4.3 transport::PublisherPtr gazebo::physics::Model::jointPub [protected]

Publisher for joint info.

The documentation for this class was generated from the following file:

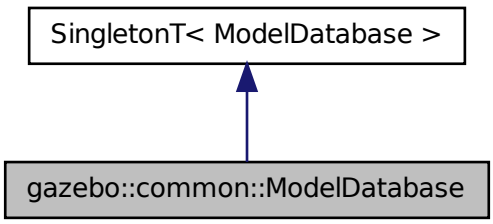
- **Model.hh**

## 10.112 gazebo::common::ModelDatabase Class Reference

Connects to model database, and has utility functions to find models.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::ModelDatabase:



### Public Member Functions

- void **DownloadDependencies** (const std::string &_path)

  *Download all dependencies for a give model path.*
- void **Fini** ()

  *Finalize the model database.*
- std::string **GetDBConfig** (const std::string &_uri)

  *Return the database.config file as a string.*
- std::string **GetModelConfig** (const std::string &_uri)

  *Return the model.config file as a string.*
- std::string **GetModelFile** (const std::string &_uri)

  *Get a model's SDF file based on a URI.*
- std::string **GetModelName** (const std::string &_uri)

  *Get the name of a model based on a URI.*
- std::string **GetModelPath** (const std::string &_uri, bool _forceDownload=false)

  *Get the local path to a model.*
- std::map< std::string, std::string > **GetModels** ()

  *Returns the dictionary of all the model names.*
- void **GetModels** (boost::function< void(const std::map< std::string, std::string > &)> _func)

  *Get the dictionary of all model names via a callback.*
- std::string **GetURI** ()

  *Returns the the global model database URI.*
- bool **HasModel** (const std::string &_modelName)

  *Returns true if the model exists on the database.*
- void **Start** (bool _fetchImmediately=false)

  *Start the model database.*

### Additional Inherited Members

#### 10.112.1 Detailed Description

Connects to model database, and has utility functions to find models.

The documentation for this class was generated from the following file:

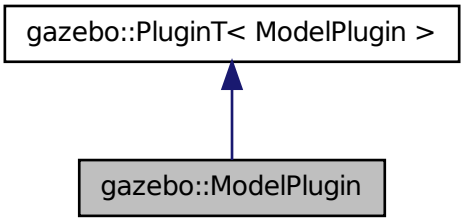
- **ModelDatabase.hh**

## 10.113 gazebo::ModelPlugin Class Reference

A plugin with access to **physics::Model** (p. 624).

`#include <Plugin.hh>`

Inheritance diagram for gazebo::ModelPlugin:




### Public Member Functions

- **ModelPlugin** ()

  *Constructor.*
- virtual ∼**ModelPlugin** ()

  *Destructor.*
- virtual void **Init** ()

  *Override this method for custom plugin initialization behavior.*
- virtual void **Load** (**physics::ModelPtr** _model, sdf::ElementPtr _sdf)=0

  *Load function.*
- virtual void **Reset** ()

  *Override this method for custom plugin reset behavior.*

### Additional Inherited Members

## 10.113.1 Detailed Description

A plugin with access to **physics::Model** (p. 624).

See `reference`.

## 10.113.2 Constructor & Destructor Documentation

#### 10.113.2.1 gazebo::ModelPlugin::ModelPlugin ( ) [inline]

Constructor.

References gazebo::MODEL_PLUGIN, and gazebo::PluginT< ModelPlugin >::type.

#### 10.113.2.2 virtual gazebo::ModelPlugin::∼ModelPlugin ( ) [inline], [virtual]

Destructor.

### 10.113.3 Member Function Documentation

#### 10.113.3.1 virtual void gazebo::ModelPlugin::Init ( ) [inline], [virtual]

Override this method for custom plugin initialization behavior.

#### 10.113.3.2 virtual void gazebo::ModelPlugin::Load ( physics::ModelPtr *_model,* sdf::ElementPtr *_sdf* ) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the Model |
| in | *_sdf* | Pointer to the SDF element of the plugin. |

#### 10.113.3.3 virtual void gazebo::ModelPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

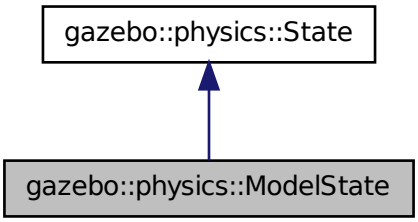
- **Plugin.hh**

## 10.114 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 624) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::ModelState:




### Public Member Functions

- **ModelState** ()

*Default constructor.*

- **ModelState** (const **ModelPtr** _model, const **common::Time** &_realTime, const **common::Time** &_simTime)

  *Constructor.*
- **ModelState** (const **ModelPtr** _model)

  *Constructor.*
- **ModelState** (const sdf::ElementPtr _sdf)

  *Constructor.*
- virtual ~**ModelState** ()

  *Destructor.*
- void **FillSDF** (sdf::ElementPtr _sdf)

  *Populate a state SDF element with data from the object.*
- **JointState GetJointState** (unsigned int _index) const

  *Get a **Joint** (p. 496) state.*
- **JointState GetJointState** (const std::string &_jointName) const

  *Get a **Joint** (p. 496) state by **Joint** (p. 496) name.*
- unsigned int **GetJointStateCount** () const

  *Get the number of joint states.*
- **JointState_M GetJointStates** (const boost::regex &_regex) const

  *Get joint states based on a regular expression.*
- const **JointState_M** & **GetJointStates** () const

  *Get the joint states.*
- **LinkState GetLinkState** (const std::string &_linkName) const

  *Get a link state by **Link** (p. 542) name.*
- unsigned int **GetLinkStateCount** () const

  *Get the number of link states.*
- **LinkState_M GetLinkStates** (const boost::regex &_regex) const

  *Get link states based on a regular expression.*
- const **LinkState_M** & **GetLinkStates** () const

  *Get the link states.*
- const **math::Pose** & **GetPose** () const

  *Get the stored model pose.*
- bool **HasJointState** (const std::string &_jointName) const

  *Return true if there is a joint with the specified name.*
- bool **HasLinkState** (const std::string &_linkName) const

  *Return true if there is a link with the specified name.*
- bool **IsZero** () const

  *Return true if the values in the state are zero.*
- void **Load** (const **ModelPtr** _model, const **common::Time** &_realTime, const **common::Time** &_simTime)

  *Load state from **Model** (p. 624) pointer.*
- virtual void **Load** (const sdf::ElementPtr _elem)

  *Load state from SDF element.*
- **ModelState operator+** (const **ModelState** &_state) const

  *Addition operator.*
- **ModelState operator-** (const **ModelState** &_state) const

  *Subtraction operator.*
- **ModelState** & **operator=** (const **ModelState** &_state)

  *Assignment operator.*

- virtual void **SetRealTime** (const **common::Time** &_time)

  *Set the real time when this state was generated.*
- virtual void **SetSimTime** (const **common::Time** &_time)

  *Set the sim time when this state was generated.*
- virtual void **SetWallTime** (const **common::Time** &_time)

  *Set the wall time when this state was generated.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::ModelState** &_state)

  *Stream insertion operator.*

## Additional Inherited Members

### 10.114.1 Detailed Description

Store state information of a **physics::Model** (p. 624) object.

This class captures the entire state of a **Model** (p. 624) at one specific time during a simulation run.

**State** (p. 998) of a **Model** (p. 624) includes the state of all its child Links and Joints.

### 10.114.2 Constructor & Destructor Documentation

#### 10.114.2.1 gazebo::physics::ModelState::ModelState ( )

Default constructor.

#### 10.114.2.2 gazebo::physics::ModelState::ModelState ( const **ModelPtr** *_model*, const **common::Time** & *_realTime*, const **common::Time** & *_simTime* )

Constructor.

Build a **ModelState** (p. 641) from an existing **Model** (p. 624).

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the model from which to gather state info. |
| in | *_realTime* | Real time stamp. |
| in | *_simTime* | Sim time stamp. |

#### 10.114.2.3 gazebo::physics::ModelState::ModelState ( const **ModelPtr** *_model* ) [explicit]

Constructor.

Build a **ModelState** (p. 641) from an existing **Model** (p. 624).

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the model from which to gather state info. |

#### 10.114.2.4 gazebo::physics::ModelState::ModelState ( const sdf::ElementPtr _sdf ) [explicit]

Constructor.

Build a **ModelState** (p. 641) from SDF data

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF data to load a model state from. |

#### 10.114.2.5 virtual gazebo::physics::ModelState::~ModelState ( ) [virtual]

Destructor.

### 10.114.3 Member Function Documentation

#### 10.114.3.1 void gazebo::physics::ModelState::FillSDF ( sdf::ElementPtr _sdf )

Populate a state SDF element with data from the object.

**Parameters**

| | | |
|---|---|---|
| out | *_sdf* | SDF element to populate. |

#### 10.114.3.2 JointState gazebo::physics::ModelState::GetJointState ( unsigned int _index ) const

Get a **Joint** (p. 496) state.

Return a **JointState** (p. 522) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 645).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of a **JointState** (p. 522). |

**Returns**

**State** (p. 998) of a **Joint** (p. 496).

**Exceptions**

| | |
|---|---|
| ***common::Exception*** *(p. 416)* | When _index is out of range. |

#### 10.114.3.3 JointState gazebo::physics::ModelState::GetJointState ( const std::string & _jointName ) const

Get a **Joint** (p. 496) state by **Joint** (p. 496) name.

Searches through all JointStates. Returns the **JointState** (p. 522) with the matching name, if any.

**Parameters**

| | | |
|---|---|---|
| in | *_jointName* | Name of the **JointState** (p. 522). |

**Returns**

**State** (p. 998) of the **Joint** (p. 496).

**Exceptions**

| | |
|---|---|
| ***common::Exception*** *(p. 416)* | When _jointName is invalid. |

#### 10.114.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount ( ) const

Get the number of joint states.

Returns the number of JointStates recorded.

**Returns**

Number of JointStates.

#### 10.114.3.5 JointState_M gazebo::physics::ModelState::GetJointStates ( const boost::regex & *_regex* ) const

Get joint states based on a regular expression.

**Parameters**

| | | |
|---|---|---|
| in | *_regex* | The regular expression. |

**Returns**

List of joint states whose names match the regular expression.

#### 10.114.3.6 const JointState_M& gazebo::physics::ModelState::GetJointStates ( ) const

Get the joint states.

**Returns**

A map of joint states.

#### 10.114.3.7 LinkState gazebo::physics::ModelState::GetLinkState ( const std::string & *_linkName* ) const

Get a link state by **Link** (p. 542) name.

Searches through all LinkStates. Returns the **LinkState** (p. 563) with the matching name, if any.

**Parameters**

| | | |
|---|---|---|
| in | *_linkName* | Name of the **LinkState** (p. 563) |

**Returns**

**State** (p. 998) of the **Link** (p. 542).

**Exceptions**

| | |
|---|---|
| ***common::Exception*** *(p. 416)* | When _linkName is invalid. |

#### 10.114.3.8 unsigned int gazebo::physics::ModelState::GetLinkStateCount ( ) const

Get the number of link states.

This returns the number of Links recorded.

**Returns**

Number of **LinkState** (p. 563) recorded.

#### 10.114.3.9 LinkState_M gazebo::physics::ModelState::GetLinkStates ( const boost::regex & *_regex* ) const

Get link states based on a regular expression.

**Parameters**

| | | |
|---|---|---|
| in | *_regex* | The regular expression. |

**Returns**

List of link states whose names match the regular expression.

#### 10.114.3.10 const LinkState_M& gazebo::physics::ModelState::GetLinkStates ( ) const

Get the link states.

**Returns**

A map of link states.

#### 10.114.3.11 const math::Pose& gazebo::physics::ModelState::GetPose ( ) const

Get the stored model pose.

**Returns**

The **math::Pose** (p. 734) of the **Model** (p. 624).

#### 10.114.3.12 bool gazebo::physics::ModelState::HasJointState ( const std::string & _jointName ) const

Return true if there is a joint with the specified name.

**Parameters**

| | | |
|---|---|---|
| in | *_jointName* | Name of the Jointtate. |

**Returns**

True if the joint exists in the model.

#### 10.114.3.13 bool gazebo::physics::ModelState::HasLinkState ( const std::string & _linkName ) const

Return true if there is a link with the specified name.

**Parameters**

| | | |
|---|---|---|
| in | *_linkName* | Name of the **LinkState** (p. 563). |

**Returns**

True if the link exists in the model.

#### 10.114.3.14 bool gazebo::physics::ModelState::IsZero ( ) const

Return true if the values in the state are zero.

**Returns**

True if the values in the state are zero.

#### 10.114.3.15 void gazebo::physics::ModelState::Load ( const ModelPtr _model, const common::Time & _realTime, const common::Time & _simTime )

Load state from **Model** (p. 624) pointer.

Build a **ModelState** (p. 641) from an existing **Model** (p. 624).

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the model from which to gather state info. |
| in | *_realTime* | Real time stamp. |
| in | *_simTime* | Sim time stamp. |

#### 10.114.3.16 virtual void gazebo::physics::ModelState::Load ( const sdf::ElementPtr _elem ) [virtual]

Load state from SDF element.

Load **ModelState** (p. 641) information from stored data in and SDF::Element

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | Pointer to the SDF::Element containing state info. |

Reimplemented from **gazebo::physics::State** (p. 1000).

#### 10.114.3.17 ModelState gazebo::physics::ModelState::operator+ ( const ModelState & *_state* ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to substract. |

**Returns**

The resulting state.

#### 10.114.3.18 ModelState gazebo::physics::ModelState::operator- ( const ModelState & *_state* ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to substract. |

**Returns**

The resulting state.

#### 10.114.3.19 ModelState& gazebo::physics::ModelState::operator= ( const ModelState & *_state* )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **State** (p. 998) value |

**Returns**

this

#### 10.114.3.20 virtual void gazebo::physics::ModelState::SetRealTime ( const common::Time & *_time* ) [virtual]

Set the real time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Clock time since simulation was stated. |

Reimplemented from **gazebo::physics::State** (p. 1001).

#### 10.114.3.21 virtual void gazebo::physics::ModelState::SetSimTime ( const common::Time & _time ) [virtual]

Set the sim time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Simulation time when the data was recorded. |

Reimplemented from **gazebo::physics::State** (p. 1001).

#### 10.114.3.22 virtual void gazebo::physics::ModelState::SetWallTime ( const common::Time & _time ) [virtual]

Set the wall time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The absolute clock time when the **State** (p. 998) data was recorded. |

Reimplemented from **gazebo::physics::State** (p. 1002).

### 10.114.4 Friends And Related Function Documentation

#### 10.114.4.1 std::ostream& operator<< ( std::ostream & _out, const gazebo::physics::ModelState & _state ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream. |
| in | *_state* | **Model** (p. 624) state to output. |

**Returns**

The stream.

The documentation for this class was generated from the following file:

- **ModelState.hh**

## 10.115 gazebo::common::MouseEvent Class Reference

Generic description of a mouse event.

`#include <common/common.hh>`

### Public Types

- enum **Buttons** { **NO_BUTTON** = 0x0, **LEFT** = 0x1, **MIDDLE** = 0x2, **RIGHT** = 0x4 }

*Standard mouse buttons enumeration.*

- enum **EventType** {
  **NO_EVENT**, **MOVE**, **PRESS**, **RELEASE**,
  **SCROLL** }

  *Mouse event types enumeration.*

### Public Member Functions

- **MouseEvent** ()

  *Constructor.*

### Public Attributes

- bool **alt**

  *Alt key press flag.*
- unsigned int **button**

  *The button which caused the event.*
- unsigned int **buttons**

  *State of the buttons when the event was generated.*
- bool **control**

  *Control key press flag.*
- bool **dragging**

  *Flag for mouse drag motion.*
- float **moveScale**

  *Scaling factor.*
- **math::Vector2i pos**

  *Mouse pointer position on the screen.*
- **math::Vector2i pressPos**

  *Position of button press.*
- **math::Vector2i prevPos**

  *Previous position.*
- **math::Vector2i scroll**

  *Scroll position.*
- bool **shift**

  *Shift key press flag.*
- **EventType type**

  *Event type.*

#### 10.115.1 Detailed Description

Generic description of a mouse event.

## 10.115.2 Member Enumeration Documentation

### 10.115.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

**Enumerator**

***NO_BUTTON***
***LEFT***
***MIDDLE***
***RIGHT***

### 10.115.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

**Enumerator**

***NO_EVENT***
***MOVE***
***PRESS***
***RELEASE***
***SCROLL***

## 10.115.3 Constructor & Destructor Documentation

### 10.115.3.1 gazebo::common::MouseEvent::MouseEvent ( ) [inline]

Constructor.

## 10.115.4 Member Data Documentation

### 10.115.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

### 10.115.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

### 10.115.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

### 10.115.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

#### 10.115.4.5 bool gazebo::common::MouseEvent::dragging

Flag for mouse drag motion.

#### 10.115.4.6 float gazebo::common::MouseEvent::moveScale

Scaling factor.

#### 10.115.4.7 math::Vector2i gazebo::common::MouseEvent::pos

Mouse pointer position on the screen.

#### 10.115.4.8 math::Vector2i gazebo::common::MouseEvent::pressPos

Position of button press.

#### 10.115.4.9 math::Vector2i gazebo::common::MouseEvent::prevPos

Previous position.

#### 10.115.4.10 math::Vector2i gazebo::common::MouseEvent::scroll

Scroll position.

#### 10.115.4.11 bool gazebo::common::MouseEvent::shift

Shift key press flag.

#### 10.115.4.12 EventType gazebo::common::MouseEvent::type

Event type.

The documentation for this class was generated from the following file:

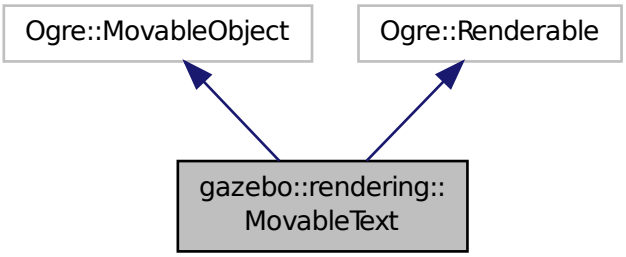
- **MouseEvent.hh**

## 10.116 gazebo::rendering::MovableText Class Reference

Movable text.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::MovableText:




## Public Types

- enum **HorizAlign** { **H_LEFT**, **H_CENTER** }

  *Horizontal alignment.*
- enum **VertAlign** { **V_BELOW**, **V_ABOVE** }

  *vertical alignment*

## Public Member Functions

- **MovableText** ()

  *Constructor.*
- virtual ~**MovableText** ()

  *Destructor.*
- **math::Box GetAABB** ()

  *Get the axis aligned bounding box of the text.*
- float **GetBaseline** () const

  *Get the baseline height.*
- float **GetCharHeight** () const

  *Set the height of a characters return Height of the characters.*
- const **common::Color** & **GetColor** () const

  *Get the text color.*
- const std::string & **GetFont** () const

  *Get the font.*
- bool **GetShowOnTop** () const

  *True = text is displayed on top.*
- float **GetSpaceWidth** () const

  *Get the width of a space.*
- const std::string & **GetText** () const

  *Get the displayed text.*
- void **Load** (const std::string &_name, const std::string &_text, const std::string &_fontName="Arial", float _charHeight=1.0, const **common::Color** &_color=**common::Color::White**)

*Loads text and font info.*

- void **SetBaseline** (float _height)

  *Set the baseline height of the text.*
- void **SetCharHeight** (float _height)

  *Set the height of a character.*
- void **SetColor** (const **common::Color** &_color)

  *Set the text color.*
- void **SetFontName** (const std::string &_font)

  *Set the font.*
- void **SetShowOnTop** (bool _show)

  *True = text always is displayed ontop.*
- void **SetSpaceWidth** (float _width)

  *Set the width of a space.*
- void **SetText** (const std::string &_text)

  *Set the text to display.*
- void **SetTextAlignment** (const **HorizAlign** &_hAlign, const **VertAlign** &_vAlign)

  *Set the alignment of the text.*
- void **Update** ()

  *Update the text.*
- virtual void **visitRenderables** (Ogre::Renderable::Visitor ∗_visitor, bool _debug=false)

### Protected Member Functions

- void **_setupGeometry** ()
- void **_updateColors** ()
- float **getBoundingRadius** () const
- const Ogre::LightList & **getLights** (void) const
- const Ogre::MaterialPtr & **getMaterial** (void) const
- void **getRenderOperation** (Ogre::RenderOperation &op)
- float **getSquaredViewDepth** (const Ogre::Camera ∗cam) const
- void **getWorldTransforms** (Ogre::Matrix4 ∗xform) const

## 10.116.1 Detailed Description

Movable text.

## 10.116.2 Member Enumeration Documentation

### 10.116.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

**Enumerator**

***H_LEFT*** Left alignment.

***H_CENTER*** Center alignment.

#### 10.116.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

**Enumerator**

***V_BELOW*** Align below.

***V_ABOVE*** Align above.

### 10.116.3 Constructor & Destructor Documentation

#### 10.116.3.1 gazebo::rendering::MovableText::MovableText ( )

Constructor.

#### 10.116.3.2 virtual gazebo::rendering::MovableText::~MovableText ( ) [virtual]

Destructor.

### 10.116.4 Member Function Documentation

#### 10.116.4.1 void gazebo::rendering::MovableText::_setupGeometry ( ) [protected]

#### 10.116.4.2 void gazebo::rendering::MovableText::_updateColors ( ) [protected]

#### 10.116.4.3 math::Box gazebo::rendering::MovableText::GetAABB ( )

Get the axis aligned bounding box of the text.

**Returns**

The axis aligned bounding box.

#### 10.116.4.4 float gazebo::rendering::MovableText::GetBaseline ( ) const

Get the baseline height.

**Returns**

Baseline height

#### 10.116.4.5 float gazebo::rendering::MovableText::getBoundingRadius ( ) const [protected]

#### 10.116.4.6 float gazebo::rendering::MovableText::GetCharHeight ( ) const

Set the height of a characters return Height of the characters.

#### 10.116.4.7 const common::Color& gazebo::rendering::MovableText::GetColor ( ) const

Get the text color.

**Returns**

Texture color.

#### 10.116.4.8 const std::string& gazebo::rendering::MovableText::GetFont ( ) const

Get the font.

**Returns**

The font name

#### 10.116.4.9 const Ogre::LightList& gazebo::rendering::MovableText::getLights ( void ) const [protected]

#### 10.116.4.10 const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial ( void ) const [protected]

#### 10.116.4.11 void gazebo::rendering::MovableText::getRenderOperation ( Ogre::RenderOperation & *op* ) [protected]

#### 10.116.4.12 bool gazebo::rendering::MovableText::GetShowOnTop ( ) const

True = text is displayed on top.

**Returns**

True if MovableText::SetShownOnTop(true) was called.

#### 10.116.4.13 float gazebo::rendering::MovableText::GetSpaceWidth ( ) const

Get the width of a space.

**Returns**

Space width

#### 10.116.4.14 float gazebo::rendering::MovableText::getSquaredViewDepth ( const Ogre::Camera * *cam* ) const [protected]

#### 10.116.4.15 const std::string& gazebo::rendering::MovableText::GetText ( ) const

Get the displayed text.

**Returns**

The displayed text.

#### 10.116.4.16 void gazebo::rendering::MovableText::getWorldTransforms ( Ogre::Matrix4 * *xform* ) const [protected]

#### 10.116.4.17 void gazebo::rendering::MovableText::Load ( const std::string & *_name*, const std::string & *_text*, const std::string & *_fontName* = "Arial", float *_charHeight* = 1.0, const common::Color & *_color* = common::Color::White )

Loads text and font info.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the text object |
| in | *_text* | Text to render |
| in | *_fontName* | Font to use |
| in | *_charHeight* | Height of the characters |
| in | *_color* | Text color |

#### 10.116.4.18 void gazebo::rendering::MovableText::SetBaseline ( float *_height* )

Set the baseline height of the text.

**Parameters**

| | | |
|---|---|---|
| in | *_height* | Baseline height |

#### 10.116.4.19 void gazebo::rendering::MovableText::SetCharHeight ( float *_height* )

Set the height of a character.

**Parameters**

| | | |
|---|---|---|
| in | *_height* | Height of the characters. |

#### 10.116.4.20 void gazebo::rendering::MovableText::SetColor ( const common::Color & *_color* )

Set the text color.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | Text color. |

#### 10.116.4.21 void gazebo::rendering::MovableText::SetFontName ( const std::string & *_font* )

Set the font.

**Parameters**

| | | |
|---|---|---|
| in | *_font* | Name of the font |

#### 10.116.4.22 void gazebo::rendering::MovableText::SetShowOnTop ( bool *_show* )

True = text always is displayed ontop.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | Set to true to render the text on top of all other drawables. |

#### 10.116.4.23 void gazebo::rendering::MovableText::SetSpaceWidth ( float *_width* )

Set the width of a space.

**Parameters**

| | | |
|---|---|---|
| in | *_width* | space width |

#### 10.116.4.24 void gazebo::rendering::MovableText::SetText ( const std::string & *_text* )

Set the text to display.

**Parameters**

| | | |
|---|---|---|
| in | *_text* | The text to display. |

#### 10.116.4.25 void gazebo::rendering::MovableText::SetTextAlignment ( const HorizAlign & *_hAlign,* const VertAlign & *_vAlign* )

Set the alignment of the text.

**Parameters**

| | | |
|---|---|---|
| in | *_hAlign* | Horizontal alignment |
| in | *_vAlign* | Vertical alignment |

#### 10.116.4.26 void gazebo::rendering::MovableText::Update ( )

Update the text.

#### 10.116.4.27 virtual void gazebo::rendering::MovableText::visitRenderables ( Ogre::Renderable::Visitor * *_visitor,* bool *_debug* = `false` ) `[virtual]`

The documentation for this class was generated from the following file:

- **MovableText.hh**

## 10.117 gazebo::msgs::MsgFactory Class Reference

A factory that generates protobuf message based on a string type.

`#include <msgs/msgs.hh>`

### Static Public Member Functions

- static void **GetMsgTypes** (std::vector< std::string > &_types)

  *Get all the message types.*
- static boost::shared_ptr < google::protobuf::Message > **NewMsg** (const std::string &_msgType)

  *Create a new instance of a message.*
- static void **RegisterMsg** (const std::string &_msgType, **MsgFactoryFn** _factoryfn)

  *Register a message.*

## 10.117.1 Detailed Description

A factory that generates protobuf message based on a string type.

## 10.117.2 Member Function Documentation

#### 10.117.2.1 static void gazebo::msgs::MsgFactory::GetMsgTypes ( std::vector< std::string > & *_types* ) [static]

Get all the message types.

**Parameters**

| | | |
|---|---|---|
| out | *_types* | Vector of strings of the message types. |

#### 10.117.2.2 static boost::shared_ptr<google::protobuf::Message> gazebo::msgs::MsgFactory::NewMsg ( const std::string & *_msgType* ) [static]

Create a new instance of a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msgType* | Type of message to create. |

**Returns**

Pointer to a google protobuf message. Null if the message type could not be handled.

#### 10.117.2.3 static void gazebo::msgs::MsgFactory::RegisterMsg ( const std::string & *_msgType*, MsgFactoryFn *_factoryfn* ) [static]

Register a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msgType* | Type of message to register. |
| in | *_factoryfn* | Function that generates the message. |

The documentation for this class was generated from the following file:

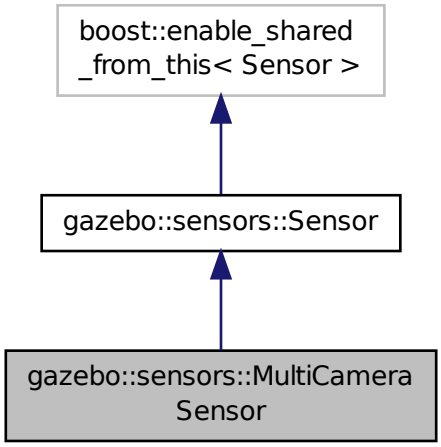
- **MsgFactory.hh**

## 10.118 gazebo::sensors::MultiCameraSensor Class Reference

Multiple camera sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::MultiCameraSensor:




### Public Member Functions

- **MultiCameraSensor** ()

  *Constructor.*
- virtual ~**MultiCameraSensor** ()

  *Destructor.*
- **rendering::CameraPtr GetCamera** (unsigned int _index) const

  *Returns a pointer to a **rendering::Camera** (p. 186).*
- unsigned int **GetCameraCount** () const

  *Get the number of cameras.*
- const unsigned char * **GetImageData** (unsigned int _index)

  *Gets the raw image data from the sensor.*
- unsigned int **GetImageHeight** (unsigned int _index) const

  *Gets the height of the image in pixels.*
- unsigned int **GetImageWidth** (unsigned int _index) const

  *Gets the width of the image in pixels.*

- virtual std::string **GetTopic** () const

  *Returns the topic name as set in SDF.*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*
- bool **SaveFrame** (const std::vector< std::string > &_filenames)

  *Saves the camera image(s) to the disk.*

### Protected Member Functions

- virtual void **Fini** ()

  *Finalize the sensor.*
- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Additional Inherited Members

### 10.118.1 Detailed Description

Multiple camera sensor.

This sensor type can create one or more synchronized cameras.

### 10.118.2 Constructor & Destructor Documentation

#### 10.118.2.1 gazebo::sensors::MultiCameraSensor::MultiCameraSensor ( )

Constructor.

#### 10.118.2.2 virtual gazebo::sensors::MultiCameraSensor::~MultiCameraSensor ( ) [virtual]

Destructor.

### 10.118.3 Member Function Documentation

#### 10.118.3.1 virtual void gazebo::sensors::MultiCameraSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.118.3.2 rendering::CameraPtr gazebo::sensors::MultiCameraSensor::GetCamera ( unsigned int _index ) const

Returns a pointer to a **rendering::Camera** (p. 186).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the camera to get |

**Returns**

The Pointer to the camera sensor.

**See Also**

**MultiCameraSensor::GetCameraCount** (p. 662)

#### 10.118.3.3 unsigned int gazebo::sensors::MultiCameraSensor::GetCameraCount ( ) const

Get the number of cameras.

**Returns**

The number of cameras.

#### 10.118.3.4 const unsigned char* gazebo::sensors::MultiCameraSensor::GetImageData ( unsigned int _index )

Gets the raw image data from the sensor.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the camera |

**Returns**

The pointer to the image data array.

**See Also**

**MultiCameraSensor::GetCameraCount** (p. 662)

#### 10.118.3.5 unsigned int gazebo::sensors::MultiCameraSensor::GetImageHeight ( unsigned int _index ) const

Gets the height of the image in pixels.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the camera |

**Returns**

The image height in pixels.

**See Also**

**MultiCameraSensor::GetCameraCount** (p. 662)

#### 10.118.3.6 unsigned int gazebo::sensors::MultiCameraSensor::GetImageWidth ( unsigned int *_index* ) const

Gets the width of the image in pixels.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the camera |

**Returns**

The image width in pixels.

**See Also**

**MultiCameraSensor::GetCameraCount** (p. 662)

#### 10.118.3.7 virtual std::string gazebo::sensors::MultiCameraSensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

**Returns**

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 843).

#### 10.118.3.8 virtual void gazebo::sensors::MultiCameraSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.118.3.9 virtual bool gazebo::sensors::MultiCameraSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.118.3.10 virtual void gazebo::sensors::MultiCameraSensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.118.3.11 bool gazebo::sensors::MultiCameraSensor::SaveFrame ( const std::vector< std::string > & *_filenames* )

Saves the camera image(s) to the disk.

**Parameters**

| | | |
|---|---|---|
| in | *_filenames* | The name of the files for each camera. |

**Returns**

True if successful, false if unsuccessful.

**See Also**

**MultiCameraSensor::GetCameraCount** (p. 662)

#### 10.118.3.12 virtual void gazebo::sensors::MultiCameraSensor::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

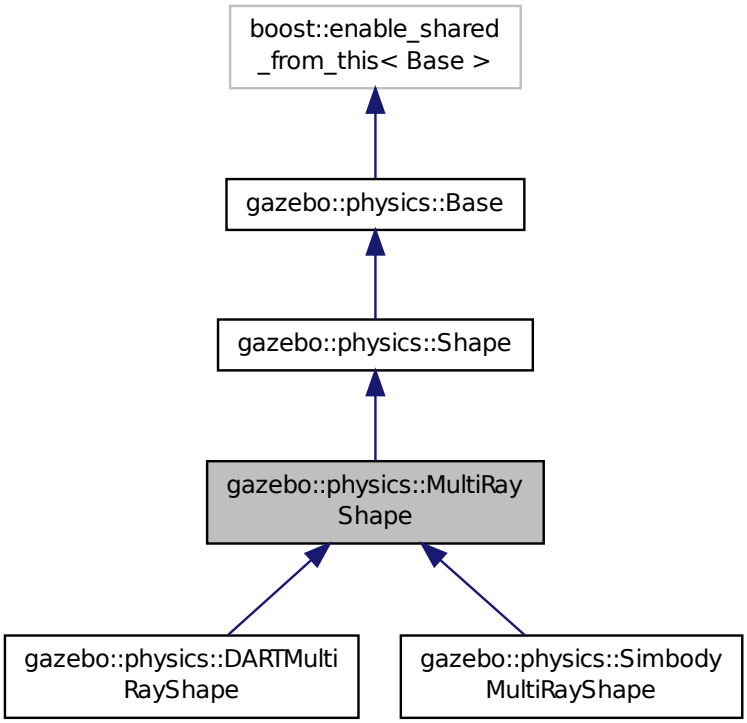
- **MultiCameraSensor.hh**

## 10.119 gazebo::physics::MultiRayShape Class Reference

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::MultiRayShape:




## Public Member Functions

- **MultiRayShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**MultiRayShape** ()

  *Destructor.*
- template<typename T >
  **event::ConnectionPtr ConnectNewLaserScans** (T _subscriber)

  *Connect a to the new laser scan signal.*
- void **DisconnectNewLaserScans** (**event::ConnectionPtr** &_conn)

  *Disconnect from the new laser scans signal.*
- void **FillMsg** (msgs::Geometry &_msg)

  *This function is not implemented.*
- int **GetFiducial** (int _index)

  *Get detected fiducial value for a ray.*
- **math::Angle GetMaxAngle** () const

  *Get the maximum angle.*
- double **GetMaxRange** () const

*Get the maximum range.*

- **math::Angle GetMinAngle** () const

  *Get the minimum angle.*

- double **GetMinRange** () const

  *Get the minimum range.*

- double **GetRange** (int _index)

  *Get detected range for a ray.*

- double **GetResRange** () const

  *Get the range resolution.*

- double **GetRetro** (int _index)

  *Get detected retro (intensity) value for a ray.*

- int **GetSampleCount** () const

  *Get the horizontal sample count.*

- double **GetScanResolution** () const

  *Get the horizontal resolution.*

- **math::Angle GetVerticalMaxAngle** () const

  *Get the vertical max angle.*

- **math::Angle GetVerticalMinAngle** () const

  *Get the vertical min angle.*

- int **GetVerticalSampleCount** () const

  *Get the vertical sample count.*

- double **GetVerticalScanResolution** () const

  *Get the vertical range resolution.*

- virtual void **Init** ()

  *Init the shape.*

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *This function is not implemented.*

- virtual void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the multi ray shape.*

- void **Update** ()

  *Update the ray collisions.*

## Protected Member Functions

- virtual void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)

  *Add a ray to the collision.*

- virtual void **UpdateRays** ()=0

  *Physics engine specific method for updating the rays.*

## Protected Attributes

- sdf::ElementPtr **horzElem**

  *Horizontal SDF element pointer.*

- **event::EventT**< void()> **newLaserScans**

  *New laser scans event.*

- **math::Pose offset**

*Pose offset of all the rays.*

- sdf::ElementPtr **rangeElem**

  *Range SDF element pointer.*

- sdf::ElementPtr **rayElem**

  *Ray SDF element pointer.*

- std::vector< **RayShapePtr** > **rays**

  *Ray data.*

- sdf::ElementPtr **scanElem**

  *Scan SDF element pointer.*

- sdf::ElementPtr **vertElem**

  *Vertical SDF element pointer.*

## Additional Inherited Members

### 10.119.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

### 10.119.2 Constructor & Destructor Documentation

#### 10.119.2.1 gazebo::physics::MultiRayShape::MultiRayShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent collision shape. |

#### 10.119.2.2 virtual gazebo::physics::MultiRayShape::~MultiRayShape ( ) [virtual]

Destructor.

### 10.119.3 Member Function Documentation

#### 10.119.3.1 virtual void gazebo::physics::MultiRayShape::AddRay ( const math::Vector3 & *_start*, const math::Vector3 & *_end* ) [protected], [virtual]

Add a ray to the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_start* | Start of the ray. |
| in | *_end* | End of the ray. |

Reimplemented in **gazebo::physics::DARTMultiRayShape** (p. 77), and **gazebo::physics::SimbodyMultiRayShape** (p. 915).

#### 10.119.3.2 template<typename T > event::ConnectionPtr gazebo::physics::MultiRayShape::ConnectNewLaserScans ( T _subscriber ) [inline]

Connect a to the new laser scan signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback function. |

**Returns**

The connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect(), and newLaserScans.

#### 10.119.3.3 void gazebo::physics::MultiRayShape::DisconnectNewLaserScans ( event::ConnectionPtr & _conn ) [inline]

Disconnect from the new laser scans signal.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | Connection to remove. |

References gazebo::event::EventT< T >::Disconnect(), and newLaserScans.

#### 10.119.3.4 void gazebo::physics::MultiRayShape::FillMsg ( msgs::Geometry & _msg ) [virtual]

This function is not implemented.

Fill a message with this shape's values.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message that contains the shape's values. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.119.3.5 int gazebo::physics::MultiRayShape::GetFiducial ( int _index )

Get detected fiducial value for a ray.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the ray. |

**Returns**

Fiducial value for the ray.

#### 10.119.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle ( ) const

Get the maximum angle.

**Returns**

Maximum angle of ray scan.

#### 10.119.3.7 double gazebo::physics::MultiRayShape::GetMaxRange ( ) const

Get the maximum range.

**Returns**

Maximum range of all the rays.

#### 10.119.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle ( ) const

Get the minimum angle.

**Returns**

Minimum angle of ray scan.

#### 10.119.3.9 double gazebo::physics::MultiRayShape::GetMinRange ( ) const

Get the minimum range.

**Returns**

Minimum range of all the rays.

#### 10.119.3.10 double gazebo::physics::MultiRayShape::GetRange ( int *_index* )

Get detected range for a ray.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the ray. |

**Returns**

Returns DBL_MAX for no detection.

#### 10.119.3.11 double gazebo::physics::MultiRayShape::GetResRange ( ) const

Get the range resolution.

**Returns**

Range resolution of all the rays.

#### 10.119.3.12 double gazebo::physics::MultiRayShape::GetRetro ( int _index )

Get detected retro (intensity) value for a ray.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the ray. |

**Returns**

Retro value for the ray.

#### 10.119.3.13 int gazebo::physics::MultiRayShape::GetSampleCount ( ) const

Get the horizontal sample count.

**Returns**

Horizontal sample count.

#### 10.119.3.14 double gazebo::physics::MultiRayShape::GetScanResolution ( ) const

Get the horizontal resolution.

**Returns**

Horizontal resolution.

#### 10.119.3.15 math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle ( ) const

Get the vertical max angle.

**Returns**

Vertical max angle.

#### 10.119.3.16 math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle ( ) const

Get the vertical min angle.

**Returns**

Vertical min angle.

#### 10.119.3.17 int gazebo::physics::MultiRayShape::GetVerticalSampleCount ( ) const

Get the vertical sample count.

**Returns**

Verical sample count.

#### 10.119.3.18 double gazebo::physics::MultiRayShape::GetVerticalScanResolution ( ) const

Get the vertical range resolution.

**Returns**

Vertical range resolution.

#### 10.119.3.19 virtual void gazebo::physics::MultiRayShape::Init ( ) [virtual]

Init the shape.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.119.3.20 virtual void gazebo::physics::MultiRayShape::ProcessMsg ( const msgs::Geometry & *_msg* ) [virtual]

This function is not implemented.

Update the ray based on a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to update from. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.119.3.21 virtual void gazebo::physics::MultiRayShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set the scale of the multi ray shape.

**Returns**

_scale Scale to set the multi ray shape to.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.119.3.22 void gazebo::physics::MultiRayShape::Update ( ) [virtual]

Update the ray collisions.

Reimplemented from **gazebo::physics::Base** (p. 171).

#### 10.119.3.23 virtual void gazebo::physics::MultiRayShape::UpdateRays ( ) [protected], [pure virtual]

Physics engine specific method for updating the rays.

Implemented in **gazebo::physics::DARTMultiRayShape** (p. 77), and **gazebo::physics::SimbodyMultiRayShape** (p. 915).

### 10.119.4 Member Data Documentation

#### 10.119.4.1 sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]

Horizontal SDF element pointer.

#### 10.119.4.2 event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]

New laser scans event.

Referenced by ConnectNewLaserScans(), and DisconnectNewLaserScans().

#### 10.119.4.3 math::Pose gazebo::physics::MultiRayShape::offset [protected]

Pose offset of all the rays.

#### 10.119.4.4 sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem [protected]

Range SDF element pointer.

#### 10.119.4.5 sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem [protected]

Ray SDF element pointer.

#### 10.119.4.6 std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays [protected]

Ray data.

#### 10.119.4.7 sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem [protected]

Scan SDF element pointer.

#### 10.119.4.8 sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem [protected]

Vertical SDF element pointer.

The documentation for this class was generated from the following file:

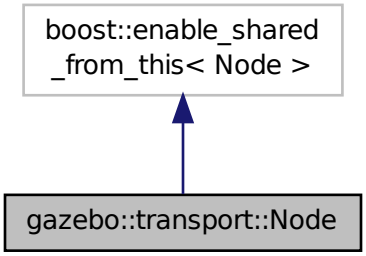
- **MultiRayShape.hh**

## 10.120 gazebo::transport::Node Class Reference

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::Node:




## Public Member Functions

- **Node** ()

  *Constructor.*
- virtual ~**Node** ()

  *Destructor.*
- template<typename M >
  **transport::PublisherPtr Advertise** (const std::string &_topic, unsigned int _queueLimit=1000, double _hzRate=0)

  *Adverise a topic.*
- std::string **DecodeTopicName** (const std::string &_topic)

  *Decode a topic name.*
- std::string **EncodeTopicName** (const std::string &_topic)

  *Encode a topic name.*
- void **Fini** ()

  *Finalize the node.*
- unsigned int **GetId** () const

  *Get the unique ID of the node.*
- std::string **GetMsgType** (const std::string &_topic) const

  *Get the message type for a topic.*
- std::string **GetTopicNamespace** () const

  *Get the topic namespace for this node.*
- bool **HandleData** (const std::string &_topic, const std::string &_msg)

  *Handle incoming data.*
- bool **HandleMessage** (const std::string &_topic, **MessagePtr** _msg)

  *Handle incoming msg.*
- bool **HasLatchedSubscriber** (const std::string &_topic) const

  *Return true if a subscriber on a specific topic is latched.*
- void **Init** (const std::string &_space="")

  *Init the node.*
- void **InsertLatchedMsg** (const std::string &_topic, const std::string &_msg)

*Add a latched message to the node for publication.*

- void **InsertLatchedMsg** (const std::string &_topic, **MessagePtr** _msg)

  *Add a latched message to the node for publication.*

- void **ProcessIncoming** ()

  *Process incoming messages.*

- void **ProcessPublishers** ()

  *Process all publishers, which has each publisher send it's most recent message over the wire.*

- template<typename M >
  void **Publish** (const std::string &_topic, const google::protobuf::Message &_message)

  *A convenience function for a one-time publication of a message.*

- void **RemoveCallback** (const std::string &_topic, unsigned int _id)

- template<typename M , typename T >
  **SubscriberPtr Subscribe** (const std::string &_topic, void(T::*_fp)(const boost::shared_ptr< M const > &), T *_obj, bool _latching=false)

  *Subscribe to a topic using a class method as the callback.*

- template<typename M >
  **SubscriberPtr Subscribe** (const std::string &_topic, void(*_fp)(const boost::shared_ptr< M const > &), bool _latching=false)

  *Subscribe to a topic using a bare function as the callback.*

- template<typename T >
  **SubscriberPtr Subscribe** (const std::string &_topic, void(T::*_fp)(const std::string &), T *_obj, bool _latching=false)

  *Subscribe to a topic using a class method as the callback.*

- **SubscriberPtr Subscribe** (const std::string &_topic, void(*_fp)(const std::string &), bool _latching=false)

  *Subscribe to a topic using a bare function as the callback.*

### 10.120.1 Detailed Description

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

### 10.120.2 Constructor & Destructor Documentation

#### 10.120.2.1 gazebo::transport::Node::Node ( )

Constructor.

#### 10.120.2.2 virtual gazebo::transport::Node::~Node ( ) [virtual]

Destructor.

### 10.120.3 Member Function Documentation

#### 10.120.3.1 template<typename M > transport::PublisherPtr gazebo::transport::Node::Advertise ( const std::string & *_topic*, unsigned int *_queueLimit* = 1000, double *_hzRate* = 0 ) [inline]

Adverise a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to advertise |
| in | *_queueLimit* | The maximum number of outgoing messages to queue for delivery |
| in | *_hz* | Update rate for the publisher. Units are 1.0/seconds. |

**Returns**

Pointer to new publisher object

References DecodeTopicName(), and SingletonT< T >::Instance().

#### 10.120.3.2 std::string gazebo::transport::Node::DecodeTopicName ( const std::string & *_topic* )

Decode a topic name.

**Parameters**

| | | |
|---|---|---|
| in | *The* | encoded name |

**Returns**

The decoded name

Referenced by Advertise(), and Subscribe().

#### 10.120.3.3 std::string gazebo::transport::Node::EncodeTopicName ( const std::string & *_topic* )

Encode a topic name.

**Parameters**

| | | |
|---|---|---|
| in | *The* | decoded name |

**Returns**

The encoded name

#### 10.120.3.4 void gazebo::transport::Node::Fini ( )

Finalize the node.

#### 10.120.3.5 unsigned int gazebo::transport::Node::GetId ( ) const

Get the unique ID of the node.

**Returns**

The unique ID of the node

Referenced by Subscribe().

#### 10.120.3.6 std::string gazebo::transport::Node::GetMsgType ( const std::string & _topic ) const

Get the message type for a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic |

**Returns**

The message type

#### 10.120.3.7 std::string gazebo::transport::Node::GetTopicNamespace ( ) const

Get the topic namespace for this node.

**Returns**

The namespace

#### 10.120.3.8 bool gazebo::transport::Node::HandleData ( const std::string & _topic, const std::string & _msg )

Handle incoming data.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Topic for which the data was received |
| in | *_msg* | The message that was received |

**Returns**

true if the message was handled successfully, false otherwise

#### 10.120.3.9 bool gazebo::transport::Node::HandleMessage ( const std::string & _topic, MessagePtr _msg )

Handle incoming msg.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Topic for which the data was received |
| in | *_msg* | The message that was received |

**Returns**

true if the message was handled successfully, false otherwise

#### 10.120.3.10 bool gazebo::transport::Node::HasLatchedSubscriber ( const std::string & _topic ) const

Return true if a subscriber on a specific topic is latched.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Name of the topic to check. |

**Returns**

True if a latched subscriber exists.

#### 10.120.3.11 void gazebo::transport::Node::Init ( const std::string & *_space* = " " )

Init the node.

**Parameters**

| | | |
|---|---|---|
| in | *_space* | Set the global namespace of all topics. If left blank, the topic will initialize to the first namespace on the **Master** (p. 583) |

#### 10.120.3.12 void gazebo::transport::Node::InsertLatchedMsg ( const std::string & *_topic,* const std::string & *_msg* )

Add a latched message to the node for publication.

This is called when a subscription is connected to a publication.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Name of the topic to publish data on. |
| in | *_msg* | The message to publish. |

#### 10.120.3.13 void gazebo::transport::Node::InsertLatchedMsg ( const std::string & *_topic,* MessagePtr *_msg* )

Add a latched message to the node for publication.

This is called when a subscription is connected to a publication.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Name of the topic to publish data on. |
| in | *_msg* | The message to publish. |

#### 10.120.3.14 void gazebo::transport::Node::ProcessIncoming ( )

Process incoming messages.

#### 10.120.3.15 void gazebo::transport::Node::ProcessPublishers ( )

Process all publishers, which has each publisher send it's most recent message over the wire.

This is for internal use only

#### 10.120.3.16 template<typename M > void gazebo::transport::Node::Publish ( const std::string & _topic, const google::protobuf::Message & _message ) [inline]

A convenience function for a one-time publication of a message.

This is inefficient, compared to **Node::Advertise** (p. 674) followed by **Publisher::Publish** (p. 759). This function should only be used when sending a message very infrequently.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to advertise |
| in | *_message* | Message to be published |

#### 10.120.3.17 void gazebo::transport::Node::RemoveCallback ( const std::string & _topic, unsigned int _id )

#### 10.120.3.18 template<typename M , typename T > SubscriberPtr gazebo::transport::Node::Subscribe ( const std::string & _topic, void(T::*)(const boost::shared_ptr< M const > &) _fp, T * _obj, bool _latching = false ) [inline]

Subscribe to a topic using a class method as the callback.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to subscribe to |
| in | *_fp* | Class method to be called on receipt of new message |
| in | *_obj* | Class instance to be used on receipt of new message |
| in | *_latching* | If true, latch latest incoming message; otherwise don't latch |

**Returns**

Pointer to new **Subscriber** (p. 1016) object

References DecodeTopicName(), GetId(), and SingletonT< T >::Instance().

#### 10.120.3.19 template<typename M > SubscriberPtr gazebo::transport::Node::Subscribe ( const std::string & _topic, void(*)(const boost::shared_ptr< M const > &) _fp, bool _latching = false ) [inline]

Subscribe to a topic using a bare function as the callback.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to subscribe to |
| in | *_fp* | Function to be called on receipt of new message |
| in | *_latching* | If true, latch latest incoming message; otherwise don't latch |

**Returns**

Pointer to new **Subscriber** (p. 1016) object

References DecodeTopicName(), GetId(), and SingletonT< T >::Instance().

#### 10.120.3.20 template<typename T > SubscriberPtr gazebo::transport::Node::Subscribe ( const std::string & _topic, void(T::*)(const std::string &) _fp, T * _obj, bool _latching = false ) [inline]

Subscribe to a topic using a class method as the callback.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to subscribe to |
| in | *_fp* | Class method to be called on receipt of new message |
| in | *_obj* | Class instance to be used on receipt of new message |
| in | *_latching* | If true, latch latest incoming message; otherwise don't latch |

**Returns**

Pointer to new **Subscriber** (p. 1016) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

#### 10.120.3.21 SubscriberPtr gazebo::transport::Node::Subscribe ( const std::string & _topic, void(*)(const std::string &) _fp, bool _latching = false ) [inline]

Subscribe to a topic using a bare function as the callback.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to subscribe to |
| in | *_fp* | Function to be called on receipt of new message |
| in | *_latching* | If true, latch latest incoming message; otherwise don't latch |

**Returns**

Pointer to new **Subscriber** (p. 1016) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

The documentation for this class was generated from the following file:

- **Node.hh**

## 10.121 gazebo::common::NodeAnimation Class Reference

Node animation.

`#include <common/common.hh>`

### Public Member Functions

- **NodeAnimation** (const std::string &_name)

  *constructor*

- ~**NodeAnimation** ()

  *Destructor. It empties the key frames list.*
- void **AddKeyFrame** (const double _time, const **math::Matrix4** _trans)

  *Adds a key frame at a specific time.*
- void **AddKeyFrame** (const double _time, const **math::Pose** _pose)

  *Adds a key fram at a specific time.*
- **math::Matrix4 GetFrameAt** (double _time, bool _loop=true) const

  *Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.*
- unsigned int **GetFrameCount** () const

  *Returns the number of key frames.*
- void **GetKeyFrame** (const unsigned int _i, double &_time, **math::Matrix4** &_trans) const

  *Finds a key frame using the index.*
- std::pair< double, **math::Matrix4** > **GetKeyFrame** (const unsigned int _i) const

  *Returns a key frame using the index.*
- double **GetLength** () const

  *Returns the duration of the animations.*
- std::string **GetName** () const

  *Returns the name.*
- double **GetTimeAtX** (const double _x) const

  *Returns the time where a transformation's translational value along the X axis is equal to _x.*
- void **Scale** (const double _scale)

  *Scales each transformation in the key frames.*
- void **SetName** (const std::string &_name)

  *Changes the name of the animation.*

## Protected Attributes

- std::map< double, **math::Matrix4** > **keyFrames**

  *the dictionary of key frames, indexed by time*
- double **length**

  *the duration of the animations (time of last key frame)*
- std::string **name**

  *the name of the animation*

### 10.121.1 Detailed Description

Node animation.

### 10.121.2 Constructor & Destructor Documentation

#### 10.121.2.1 gazebo::common::NodeAnimation::NodeAnimation ( const std::string & *_name* )

constructor

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the node |

#### 10.121.2.2 gazebo::common::NodeAnimation::~NodeAnimation ( )

Destructor. It empties the key frames list.

### 10.121.3 Member Function Documentation

#### 10.121.3.1 void gazebo::common::NodeAnimation::AddKeyFrame ( const double *_time,* const math::Matrix4 *_trans* )

Adds a key frame at a specific time.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time of the key frame |
| in | *_trans* | the transformation |

#### 10.121.3.2 void gazebo::common::NodeAnimation::AddKeyFrame ( const double *_time,* const math::Pose *_pose* )

Adds a key fram at a specific time.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the tiem of the key frame |
| in | *_pose* | the pose |

#### 10.121.3.3 math::Matrix4 gazebo::common::NodeAnimation::GetFrameAt ( double *_time,* bool *_loop* = `true` ) const

Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time |
| in | *_loop* | when true, the time is divided by the duration (see GetLength) |

#### 10.121.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount ( ) const

Returns the number of key frames.

**Returns**

the count

#### 10.121.3.5 void gazebo::common::NodeAnimation::GetKeyFrame ( const unsigned int *_i,* double & *_time,* math::Matrix4 & *_trans* ) const

Finds a key frame using the index.

Note the index of a key frame can change as frames are added.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | the index |
| out | *_time* | the time of the frame, or -1 if the index id is out of bounds |
| out | *_trans* | the transformation for this key frame |

#### 10.121.3.6 std::pair<double, math::Matrix4> gazebo::common::NodeAnimation::GetKeyFrame ( const unsigned int _i ) const

Returns a key frame using the index.

Note the index of a key frame can change as frames are added.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | the index |

**Returns**

a pair that contains the time and transformation. **Time** (p. 1031) is -1 if the index is out of bounds

#### 10.121.3.7 double gazebo::common::NodeAnimation::GetLength ( ) const

Returns the duration of the animations.

**Returns**

the time of the last animation

#### 10.121.3.8 std::string gazebo::common::NodeAnimation::GetName ( ) const

Returns the name.

**Returns**

the name

#### 10.121.3.9 double gazebo::common::NodeAnimation::GetTimeAtX ( const double _x ) const

Returns the time where a transformation's translational value along the X axis is equal to _x.

When no transformation is found (within a tolerance of 1e-6), the time is interpolated.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | the value along x. You must ensure that _x is within a valid range. |

#### 10.121.3.10 void gazebo::common::NodeAnimation::Scale ( const double _scale )

Scales each transformation in the key frames.

This only affects the translational values.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | the scaling factor |

#### 10.121.3.11 void gazebo::common::NodeAnimation::SetName ( const std::string & *_name* )

Changes the name of the animation.

**Parameters**

| | | |
|---|---|---|
| in | *the* | new name |

### 10.121.4 Member Data Documentation

#### 10.121.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

#### 10.121.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

#### 10.121.4.3 std::string gazebo::common::NodeAnimation::name [protected]

the name of the animation

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

## 10.122 gazebo::common::NodeAssignment Struct Reference

Vertex to node weighted assignement for skeleton animation visualization.

`#include <Mesh.hh>`

### Public Attributes

- unsigned int **nodeIndex**

  *node (or bone) index*
- unsigned int **vertexIndex**

  *index of the vertex*
- float **weight**

  *the weight (between 0 and 1)*

### 10.122.1 Detailed Description

Vertex to node weighted assignement for skeleton animation visualization.

### 10.122.2 Member Data Documentation

#### 10.122.2.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

#### 10.122.2.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

#### 10.122.2.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

The documentation for this struct was generated from the following file:

- **Mesh.hh**

## 10.123 gazebo::common::NodeTransform Class Reference

**NodeTransform** (p. 684) **Skeleton.hh** (p. 1367) common/common.hh

```
#include <Skeleton.hh>
```

### Public Types

- enum **TransformType** { **TRANSLATE**, **ROTATE**, **SCALE**, **MATRIX** }

  *Enumeration of the transform types.*

### Public Member Functions

- **NodeTransform** (**TransformType** _type=**MATRIX**)

  *Constructor.*
- **NodeTransform** (**math::Matrix4** _mat, std::string _sid="_default_", TransformType _type=**MATRIX**)

  *Constructor.*
- ~**NodeTransform** ()

  *Destructor. It does nothing.*
- **math::Matrix4 Get** ()

  *Returns the transformation matrix.*
- std::string **GetSID** ()

  *Returns thr SID.*
- **TransformType GetType** ()

  *Returns the transformation type.*

- **math::Matrix4 operator()** ()

  *Matrix cast operator.*
- **math::Matrix4 operator*** (**NodeTransform** _t)

  *Node transform multiplication operator.*
- **math::Matrix4 operator*** (**math::Matrix4** _m)

  *Matrix multiplication operator.*
- void **PrintSource** ()

  *Prints the transform matrix to std::err stream.*
- void **RecalculateMatrix** ()

  *Sets the transform matrix from the source according to the type.*
- void **Set** (**math::Matrix4** _mat)

  *Assign a transformation.*
- void **SetComponent** (unsigned int _idx, double _value)

  *Set a transformation matrix component value.*
- void **SetSID** (std::string _sid)

  *Set the SID.*
- void **SetSourceValues** (**math::Matrix4** _mat)

  *Set source data values _ param[in] _mat the values.*
- void **SetSourceValues** (**math::Vector3** _vec)

  *Set source data values.*
- void **SetSourceValues** (**math::Vector3** _axis, double _angle)

  *Sets source matrix values from roation.*
- void **SetType** (**TransformType** _type)

  *Set transform type.*

## Protected Attributes

- std::string **sid**

  *the sid*
- std::vector< double > **source**

  *source data values (can be a matrix, a position or rotation)*
- **math::Matrix4 transform**

  *transform*
- **TransformType type**

  *transform type*

### 10.123.1 Detailed Description

**NodeTransform** (p. 684) **Skeleton.hh** (p. 1367) common/common.hh

A transformation node

### 10.123.2 Member Enumeration Documentation

#### 10.123.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

**Enumerator**

***TRANSLATE***

***ROTATE***

***SCALE***

***MATRIX***

### 10.123.3 Constructor & Destructor Documentation

#### 10.123.3.1 gazebo::common::NodeTransform::NodeTransform ( TransformType *_type* = MATRIX )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | the type of transform |

#### 10.123.3.2 gazebo::common::NodeTransform::NodeTransform ( math::Matrix4 *_mat*, std::string *_sid* = "_default_", TransformType *_type* = MATRIX )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_mat* | the matrix |
| in | *_sid* | identifier |
| in | *_type* | the type of transform |

#### 10.123.3.3 gazebo::common::NodeTransform::~NodeTransform ( )

Destructor. It does nothing.

### 10.123.4 Member Function Documentation

#### 10.123.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ( )

Returns the transformation matrix.

**Returns**

the matrix

#### 10.123.4.2 std::string gazebo::common::NodeTransform::GetSID ( )

Returns thr SID.

**Returns**

the SID

#### 10.123.4.3 TransformType gazebo::common::NodeTransform::GetType ( )

Returns the transformation type.

**Returns**

the type

#### 10.123.4.4 math::Matrix4 gazebo::common::NodeTransform::operator() ( )

Matrix cast operator.

**Returns**

the transform

#### 10.123.4.5 math::Matrix4 gazebo::common::NodeTransform::operator* ( NodeTransform _t )

Node transform multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_t* | a transform |

**Returns**

transform matrix multiplied by _t's transform

#### 10.123.4.6 math::Matrix4 gazebo::common::NodeTransform::operator* ( math::Matrix4 _m )

Matrix multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_m* | a matrix |

**Returns**

transform matrix multiplied by _m

#### 10.123.4.7 void gazebo::common::NodeTransform::PrintSource ( )

Prints the transform matrix to std::err stream.

#### 10.123.4.8 void gazebo::common::NodeTransform::RecalculateMatrix ( )

Sets the transform matrix from the source according to the type.

#### 10.123.4.9 void gazebo::common::NodeTransform::Set ( math::Matrix4 *_mat* )

Assign a transformation.

**Parameters**

| | | |
|---|---|---|
| in | *_mat* | the transform |

#### 10.123.4.10 void gazebo::common::NodeTransform::SetComponent ( unsigned int *_idx*, double *_value* )

Set a transformation matrix component value.

**Parameters**

| | | |
|---|---|---|
| in | *_idx* | the component index |
| in | *_value* | the value |

#### 10.123.4.11 void gazebo::common::NodeTransform::SetSID ( std::string *_sid* )

Set the SID.

**Parameters**

| | | |
|---|---|---|
| in | *_sid* | the sid |

#### 10.123.4.12 void gazebo::common::NodeTransform::SetSourceValues ( math::Matrix4 *_mat* )

Set source data values _ param[in] _mat the values.

#### 10.123.4.13 void gazebo::common::NodeTransform::SetSourceValues ( math::Vector3 *_vec* )

Set source data values.

#### 10.123.4.14 void gazebo::common::NodeTransform::SetSourceValues ( math::Vector3 *_axis*, double *_angle* )

Sets source matrix values from roation.

**Parameters**

| | | |
|---|---|---|
| in | *_axis* | of rotation |
| in | *_angle* | of rotation |

#### 10.123.4.15 void gazebo::common::NodeTransform::SetType ( TransformType _type )

Set transform type.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | the type |

### 10.123.5 Member Data Documentation

#### 10.123.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

#### 10.123.5.2 std::vector<double> gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

#### 10.123.5.3 math::Matrix4 gazebo::common::NodeTransform::transform [protected]

transform

#### 10.123.5.4 TransformType gazebo::common::NodeTransform::type [protected]

transform type

The documentation for this class was generated from the following file:

- **Skeleton.hh**

## 10.124 gazebo::sensors::Noise Class Reference

**Noise** (p. 689) models for sensor output signals.

`#include <sensors/sensors.hh>`

### Public Types

- enum **NoiseType** { **NONE**, **GAUSSIAN**, **GAUSSIAN_QUANTIZED** }
  *Which noise types we support.*

### Public Member Functions

- **Noise** ()
  *Constructor.*
- ~**Noise** ()
  *Destructor.*
- double **Apply** (double _in) const

*Apply noise to input data value.*

- double **GetBias** () const

  *Accessor for bias.*

- double **GetMean** () const

  *Accessor for mean.*

- **NoiseType GetNoiseType** () const

  *Accessor for NoiseType.*

- double **GetStdDev** () const

  *Accessor for stddev.*

- void **Load** (sdf::ElementPtr _sdf)

  *Load parameters from sdf.*

### 10.124.1 Detailed Description

**Noise** (p. 689) models for sensor output signals.

### 10.124.2 Member Enumeration Documentation

#### 10.124.2.1 enum gazebo::sensors::Noise::NoiseType

Which noise types we support.

**Enumerator**

***NONE***

***GAUSSIAN***

***GAUSSIAN_QUANTIZED***

### 10.124.3 Constructor & Destructor Documentation

#### 10.124.3.1 gazebo::sensors::Noise::Noise ( )

Constructor.

#### 10.124.3.2 gazebo::sensors::Noise::~Noise ( )

Destructor.

### 10.124.4 Member Function Documentation

#### 10.124.4.1 double gazebo::sensors::Noise::Apply ( double *_in* ) const

Apply noise to input data value.

**Parameters**

| | | |
|---|---|---|
| in | *_in* | Input data value. |

**Returns**

Data with noise applied.

#### 10.124.4.2 double gazebo::sensors::Noise::GetBias ( ) const

Accessor for bias.

**Returns**

Bias on output.

#### 10.124.4.3 double gazebo::sensors::Noise::GetMean ( ) const

Accessor for mean.

**Returns**

Mean of Gaussian noise.

#### 10.124.4.4 NoiseType gazebo::sensors::Noise::GetNoiseType ( ) const

Accessor for NoiseType.

**Returns**

Type of noise currently in use.

#### 10.124.4.5 double gazebo::sensors::Noise::GetStdDev ( ) const

Accessor for stddev.

**Returns**

Standard deviation of Gaussian noise.

#### 10.124.4.6 void gazebo::sensors::Noise::Load ( sdf::ElementPtr *_sdf* )

Load parameters from sdf.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters. |

The documentation for this class was generated from the following file:

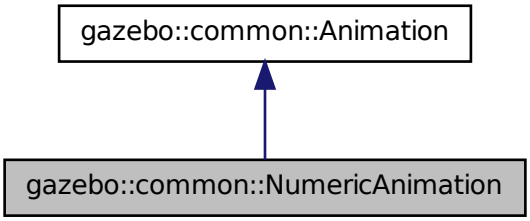
- **Noise.hh**

## 10.125 gazebo::common::NumericAnimation Class Reference

A numeric animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::NumericAnimation:




### Public Member Functions

- **NumericAnimation** (const std::string &_name, double _length, bool _loop)

  *Constructor.*
- virtual ~**NumericAnimation** ()

  *Destructor.*
- **NumericKeyFrame** * **CreateKeyFrame** (double _time)

  *Create a numeric keyframe at the given time.*
- void **GetInterpolatedKeyFrame** (**NumericKeyFrame** &_kf) const

  *Get a keyframe using the animation's current time.*

### Additional Inherited Members

### 10.125.1 Detailed Description

A numeric animation.

### 10.125.2 Constructor & Destructor Documentation

#### 10.125.2.1 gazebo::common::NumericAnimation::NumericAnimation ( const std::string & *_name,* double *_length,* bool *_loop* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | String name of the animation. This should be unique. |
| in | *_length* | Length of the animation in seconds |
| in | *_loop* | True == loop the animation |

#### 10.125.2.2 virtual gazebo::common::NumericAnimation::~NumericAnimation ( ) [virtual]

Destructor.

### 10.125.3 Member Function Documentation

#### 10.125.3.1 NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame ( double *_time* )

Create a numeric keyframe at the given time.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | **Time** (p. 1031) at which to create the keyframe |

**Returns**

Pointer to the new keyframe

#### 10.125.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame ( NumericKeyFrame & *_kf* ) const

Get a keyframe using the animation's current time.

**Parameters**

| | | |
|---|---|---|
| out | *_kf* | **NumericKeyFrame** (p. 693) reference to hold the interpolated result |

The documentation for this class was generated from the following file:

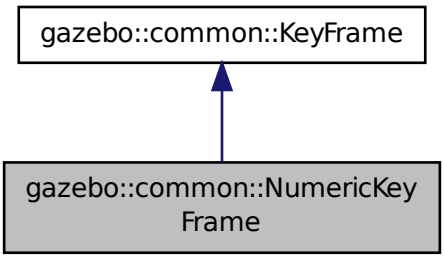
- **Animation.hh**

## 10.126 gazebo::common::NumericKeyFrame Class Reference

A keyframe for a **NumericAnimation** (p. 692).

`#include <KeyFrame.hh>`

Inheritance diagram for gazebo::common::NumericKeyFrame:




### Public Member Functions

- **NumericKeyFrame** (double _time)

  *Constructor.*
- virtual ~**NumericKeyFrame** ()

  *Destructor.*
- const double & **GetValue** () const

  *Get the value of the keyframe.*
- void **SetValue** (const double &_value)

  *Set the value of the keyframe.*

### Protected Attributes

- double **value**

  *numeric value*

## 10.126.1 Detailed Description

A keyframe for a **NumericAnimation** (p. 692).

## 10.126.2 Constructor & Destructor Documentation

#### 10.126.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame ( double *_time* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | **Time** (p. 1031) of the keyframe |

#### 10.126.2.2 virtual gazebo::common::NumericKeyFrame::~NumericKeyFrame ( ) [virtual]

Destructor.

### 10.126.3 Member Function Documentation

#### 10.126.3.1 const double& gazebo::common::NumericKeyFrame::GetValue ( ) const

Get the value of the keyframe.

**Returns**

the value of the keyframe

#### 10.126.3.2 void gazebo::common::NumericKeyFrame::SetValue ( const double & *_value* )

Set the value of the keyframe.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | The new value |

### 10.126.4 Member Data Documentation

#### 10.126.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

The documentation for this class was generated from the following file:

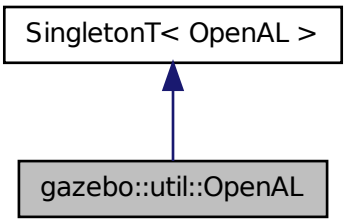
- **KeyFrame.hh**

## 10.127 gazebo::util::OpenAL Class Reference

3D audio setup and playback.

`#include <util/util.hh>`

Inheritance diagram for gazebo::util::OpenAL:




## Public Member Functions

- **OpenALSinkPtr CreateSink** (sdf::ElementPtr _sdf)

  *Create an audio listener.*
- **OpenALSourcePtr CreateSource** (sdf::ElementPtr _sdf)

  *Create an **OpenALSource** (p. 698) object.*
- void **Fini** ()

  *Finalize.*
- bool **Load** (sdf::ElementPtr _sdf=sdf::ElementPtr())

  *Load the **OpenAL** (p. 695) server.*

## Additional Inherited Members

### 10.127.1 Detailed Description

3D audio setup and playback.

### 10.127.2 Member Function Documentation

#### 10.127.2.1 OpenALSinkPtr gazebo::util::OpenAL::CreateSink ( sdf::ElementPtr *_sdf* )

Create an audio listener.

Currenly, only one listener may be created.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF element parameters for an audio_source. |

**Returns**

A pointer to an **OpenALSink** (p. 697) object.

#### 10.127.2.2 OpenALSourcePtr gazebo::util::OpenAL::CreateSource ( sdf::ElementPtr _sdf )

Create an **OpenALSource** (p. 698) object.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF element parameters for an audio_source. |

**Returns**

A pointer to an **OpenALSource** (p. 698) object.

#### 10.127.2.3 void gazebo::util::OpenAL::Fini ( )

Finalize.

#### 10.127.2.4 bool gazebo::util::OpenAL::Load ( sdf::ElementPtr *_sdf* = `sdf::ElementPtr()` )

Load the **OpenAL** (p. 695) server.

**Returns**

True on success.

The documentation for this class was generated from the following file:

- **OpenAL.hh**

## 10.128 gazebo::util::OpenALSink Class Reference

**OpenAL** (p. 695) Listener.

`#include <OpenAL.hh>`

### Public Member Functions

- **OpenALSink** ()
  *Constructor.*
- virtual ~**OpenALSink** ()
  *Destructor.*
- bool **SetPose** (const **math::Pose** &_pose)
  *Set the position of the sink.*
- bool **SetVelocity** (const **math::Vector3** &_vel)
  *Set the velocity of the sink.*

### 10.128.1 Detailed Description

**OpenAL** (p. 695) Listener.

This can be thought of as a microphone.

### 10.128.2 Constructor & Destructor Documentation

#### 10.128.2.1 gazebo::util::OpenALSink::OpenALSink ( )

Constructor.

#### 10.128.2.2 virtual gazebo::util::OpenALSink::~OpenALSink ( ) [virtual]

Destructor.

### 10.128.3 Member Function Documentation

#### 10.128.3.1 bool gazebo::util::OpenALSink::SetPose ( const math::Pose & _pose )

Set the position of the sink.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | New pose of the sink. |

**Returns**

True on success.

#### 10.128.3.2 bool gazebo::util::OpenALSink::SetVelocity ( const math::Vector3 & _vel )

Set the velocity of the sink.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Velocity of the sink. |

**Returns**

True on success.

The documentation for this class was generated from the following file:

- **OpenAL.hh**

## 10.129 gazebo::util::OpenALSource Class Reference

**OpenAL** (p. 695) Source.

```
#include <OpenAL.hh>
```

### Public Member Functions

- **OpenALSource** ()

*Constructor.*

- virtual ~**OpenALSource** ()

  *Destructor.*
- void **FillBufferFromFile** (const std::string &_audioFile)

  *Fill the **OpenAL** (p. 695) audio buffer with data from a sound file.*
- bool **FillBufferFromPCM** (uint8_t *_pcmData, unsigned int _dataCount, int _sampleRate)

  *Fill the **OpenAL** (p. 695) audio buffer from PCM data.*
- std::vector< std::string > **GetCollisionNames** () const

  *Get a vector of all the collision names.*
- bool **GetOnContact** () const

  *Return true if the audio source is played on contact with another object.*
- bool **HasCollisionName** (const std::string &_name) const

  *Get whether the source has a collision name set.*
- bool **IsPlaying** ()

  *Is the audio playing.*
- bool **Load** (sdf::ElementPtr _sdf)

  *Load the source from sdf.*
- void **Pause** ()

  *Pause a sound.*
- void **Play** ()

  *Play a sound.*
- void **Rewind** ()

  *Rewind the sound to the beginning.*
- bool **SetGain** (float _g)

  *Set the pitch of the source.*
- bool **SetLoop** (bool _state)

  *Set whether the source loops the audio.*
- bool **SetPitch** (float _p)

  *Set the pitch of the source.*
- bool **SetPose** (const **math::Pose** &_pose)

  *Set the position of the source.*
- bool **SetVelocity** (const **math::Vector3** &_vel)

  *Set the velocity of the source.*
- void **Stop** ()

  *Stop a sound.*

## 10.129.1 Detailed Description

**OpenAL** (p. 695) Source.

This can be thought of as a speaker.

## 10.129.2 Constructor & Destructor Documentation

### 10.129.2.1 gazebo::util::OpenALSource::OpenALSource ( )

Constructor.

#### 10.129.2.2 virtual gazebo::util::OpenALSource::~OpenALSource ( ) [virtual]

Destructor.

### 10.129.3 Member Function Documentation

#### 10.129.3.1 void gazebo::util::OpenALSource::FillBufferFromFile ( const std::string & *_audioFile* )

Fill the **OpenAL** (p. 695) audio buffer with data from a sound file.

**Parameters**

| | | |
|---|---|---|
| in | *_audioFile* | Name and an audio file. |

#### 10.129.3.2 bool gazebo::util::OpenALSource::FillBufferFromPCM ( uint8_t * *_pcmData,* unsigned int *_dataCount,* int *_sampleRate* )

Fill the **OpenAL** (p. 695) audio buffer from PCM data.

**Parameters**

| | | |
|---|---|---|
| in | *_pcmData* | Pointer to the PCM audio data. |
| in | *_dataCount* | Size of the PCM data. |
| in | *_sampleRate* | Sample rate for the PCM data. |

**Returns**

True on success.

#### 10.129.3.3 std::vector<std::string> gazebo::util::OpenALSource::GetCollisionNames ( ) const

Get a vector of all the collision names.

**Returns**

All the collision names used to trigger audio playback on contact.

#### 10.129.3.4 bool gazebo::util::OpenALSource::GetOnContact ( ) const

Return true if the audio source is played on contact with another object.

Contact is determine based on a set of collision objects.

**Returns**

True if audio is played on contact.

**See Also**

AddCollision()

#### 10.129.3.5 bool gazebo::util::OpenALSource::HasCollisionName ( const std::string & _name ) const

Get whether the source has a collision name set.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of a collision to check for. |

**Returns**

True if the collision name was found.

#### 10.129.3.6 bool gazebo::util::OpenALSource::IsPlaying ( )

Is the audio playing.

#### 10.129.3.7 bool gazebo::util::OpenALSource::Load ( sdf::ElementPtr _sdf )

Load the source from sdf.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF element parameters for an audio_source. |

**Returns**

True on success.

#### 10.129.3.8 void gazebo::util::OpenALSource::Pause ( )

Pause a sound.

#### 10.129.3.9 void gazebo::util::OpenALSource::Play ( )

Play a sound.

#### 10.129.3.10 void gazebo::util::OpenALSource::Rewind ( )

Rewind the sound to the beginning.

#### 10.129.3.11 bool gazebo::util::OpenALSource::SetGain ( float _g )

Set the pitch of the source.

**Parameters**

| | | |
|---|---|---|
| in | *_g* | Gain value. |

**Returns**

True on success.

#### 10.129.3.12 bool gazebo::util::OpenALSource::SetLoop ( bool *_state* )

Set whether the source loops the audio.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | True to cause playback to loop. |

**Returns**

True on success.

#### 10.129.3.13 bool gazebo::util::OpenALSource::SetPitch ( float *_p* )

Set the pitch of the source.

**Parameters**

| | | |
|---|---|---|
| in | *_p* | Pitch value. |

**Returns**

True on success.

#### 10.129.3.14 bool gazebo::util::OpenALSource::SetPose ( const **math::Pose** & *_pose* )

Set the position of the source.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | New pose of the source. |

**Returns**

True on success.

#### 10.129.3.15 bool gazebo::util::OpenALSource::SetVelocity ( const **math::Vector3** & *_vel* )

Set the velocity of the source.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | New velocity of the source. |

**Returns**

True on success.

#### 10.129.3.16 void gazebo::util::OpenALSource::Stop ( )

Stop a sound.

The documentation for this class was generated from the following file:

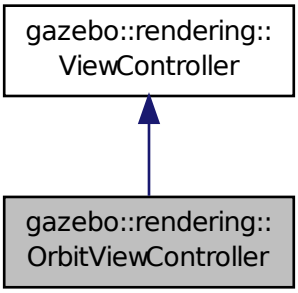
- **OpenAL.hh**

## 10.130 gazebo::rendering::OrbitViewController Class Reference

Orbit view controller.

`#include <OrbitViewController.hh>`

Inheritance diagram for gazebo::rendering::OrbitViewController:




### Public Member Functions

- **OrbitViewController** (**UserCameraPtr** _camera)

  *Constructor.*
- virtual ~**OrbitViewController** ()

  *Destructor.*
- **math::Vector3 GetFocalPoint** () const

  *Get the focal point.*
- virtual void **HandleKeyPressEvent** (const std::string &_key)

  *Handle a key press event.*
- void **HandleKeyReleaseEvent** (const std::string &_key)

  *Handle a key release event.*
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)

*Handle a mouse event.*

- virtual void **Init** ()

  *Initialize the controller.*

- virtual void **Init** (const **math::Vector3** &_focalPoint)

  *Initialze the controller with a focal point.*

- void **SetDistance** (float _d)

  *Set the distance to the focal point.*

- void **SetFocalPoint** (const **math::Vector3** &_fp)

  *Set the focal point.*

- virtual void **Update** ()

  *Update.*

### Static Public Member Functions

- static std::string **GetTypeString** ()

  *Get the type name of this view controller.*

### Additional Inherited Members

## 10.130.1 Detailed Description

Orbit view controller.

## 10.130.2 Constructor & Destructor Documentation

### 10.130.2.1 gazebo::rendering::OrbitViewController::OrbitViewController ( UserCameraPtr *_camera* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | Pointer to the camera to control. |

### 10.130.2.2 virtual gazebo::rendering::OrbitViewController::~OrbitViewController ( ) [virtual]

Destructor.

## 10.130.3 Member Function Documentation

### 10.130.3.1 math::Vector3 gazebo::rendering::OrbitViewController::GetFocalPoint ( ) const

Get the focal point.

**Returns**

The focal point

#### 10.130.3.2 static std::string gazebo::rendering::OrbitViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

**Returns**

The view controller name: "orbit".

#### 10.130.3.3 virtual void gazebo::rendering::OrbitViewController::HandleKeyPressEvent ( const std::string & _key ) [virtual]

Handle a key press event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key that was pressed. |

Implements **gazebo::rendering::ViewController** (p. 1119).

#### 10.130.3.4 void gazebo::rendering::OrbitViewController::HandleKeyReleaseEvent ( const std::string & _key ) [virtual]

Handle a key release event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key that was released. |

Implements **gazebo::rendering::ViewController** (p. 1120).

#### 10.130.3.5 virtual void gazebo::rendering::OrbitViewController::HandleMouseEvent ( const common::MouseEvent & _event ) [virtual]

Handle a mouse event.

**Parameters**

| | | |
|---|---|---|
| in | *_event* | The mouse event. |

Implements **gazebo::rendering::ViewController** (p. 1120).

#### 10.130.3.6 virtual void gazebo::rendering::OrbitViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 1120).

#### 10.130.3.7 virtual void gazebo::rendering::OrbitViewController::Init ( const math::Vector3 & _focalPoint ) [virtual]

Initialze the controller with a focal point.

**Parameters**

| | | |
|---|---|---|
| in | *_focalPoint* | Point to look at. |

Reimplemented from **gazebo::rendering::ViewController** (p. 1120).

#### 10.130.3.8 void gazebo::rendering::OrbitViewController::SetDistance ( float _d )

Set the distance to the focal point.

**Parameters**

| | | |
|---|---|---|
| in | *_d* | The distance from the focal point. |

#### 10.130.3.9 void gazebo::rendering::OrbitViewController::SetFocalPoint ( const math::Vector3 & _fp )

Set the focal point.

**Parameters**

| | | |
|---|---|---|
| in | *_fp* | The focal point |

#### 10.130.3.10 virtual void gazebo::rendering::OrbitViewController::Update ( ) [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 1121).

The documentation for this class was generated from the following file:

- **OrbitViewController.hh**

## 10.131 gazebo::common::ParamT< T > Class Template Reference

`#include <CommonTypes.hh>`

The documentation for this class was generated from the following file:

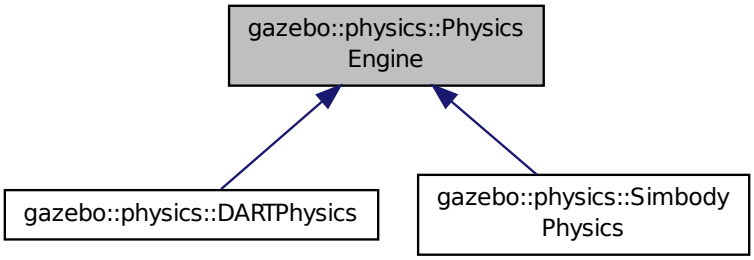
- **CommonTypes.hh**

## 10.132 gazebo::physics::PhysicsEngine Class Reference

**Base** (p. 159) class for a physics engine.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::PhysicsEngine:




## Public Member Functions

- **PhysicsEngine** (**WorldPtr** _world)

  *Default constructor.*
- virtual ~**PhysicsEngine** ()

  *Destructor.*
- virtual **CollisionPtr CreateCollision** (const std::string &_shapeType, **LinkPtr** _link)=0

  *Create a collision.*
- **CollisionPtr CreateCollision** (const std::string &_shapeType, const std::string &_linkName)

  *Create a collision.*
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent=**ModelPtr**())=0

  *Create a new joint.*
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)=0

  *Create a new body.*
- virtual **ModelPtr CreateModel** (**BasePtr** _base)

  *Create a new model.*
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)=0

  *Create a **physics::Shape** (p. 861) object.*
- virtual void **DebugPrint** () const =0

  *Debug print out of the physic engine state.*
- virtual void **Fini** ()

  *Finilize the physics engine.*
- virtual bool **GetAutoDisableFlag** ()

  *: Remove this function, and replace it with a more generic property map*
- **ContactManager** * **GetContactManager** () const

  *Get a pointer to the contact manger.*
- virtual double **GetContactMaxCorrectingVel** ()

  *: Remove this function, and replace it with a more generic property map.*
- virtual double **GetContactSurfaceLayer** ()

  *: Remove this function, and replace it with a more generic property map.*
- virtual **math::Vector3 GetGravity** () const

*Return the gavity vector.*

- virtual int **GetMaxContacts** ()

  *: Remove this function, and replace it with a more generic property map.*

- double **GetMaxStepSize** () const

  *Get max step size.*

- virtual boost::any **GetParam** (std::string _key) const

  *Get an parameter of the physics engine.*

- boost::recursive_mutex * **GetPhysicsUpdateMutex** () const

  *returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 720).*

- double **GetRealTimeUpdateRate** () const

  *Get real time update rate.*

- virtual int **GetSORPGSIters** ()

  *: Remove this function, and replace it with a more generic property map*

- virtual int **GetSORPGSPreconIters** ()

  *: Remove this function, and replace it with a more generic property map*

- virtual double **GetSORPGSW** ()

  *: Remove this function, and replace it with a more generic property map.*

- double **GetTargetRealTimeFactor** () const

  *Get target real time factor.*

- virtual std::string **GetType** () const =0

  *Return the physics engine type (ode|bullet|dart|simbody).*

- double **GetUpdatePeriod** ()

  *Get the simulation update period.*

- virtual double **GetWorldCFM** ()

  *: Remove this function, and replace it with a more generic property map*

- virtual double **GetWorldERP** ()

  *: Remove this function, and replace it with a more generic property map*

- virtual void **Init** ()=0

  *Initialize the physics engine.*

- virtual void **InitForThread** ()=0

  *Init the engine for threads.*

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the physics engine.*

- virtual void **Reset** ()

  *Rest the physics engine.*

- virtual void **SetAutoDisableFlag** (bool _autoDisable)

  *: Remove this function, and replace it with a more generic property map*

- virtual void **SetContactMaxCorrectingVel** (double _vel)

  *: Remove this function, and replace it with a more generic property map*

- virtual void **SetContactSurfaceLayer** (double _layerDepth)

  *: Remove this function, and replace it with a more generic property map*

- virtual void **SetGravity** (const **gazebo::math::Vector3** &_gravity)=0

  *Set the gavity vector.*

- virtual void **SetMaxContacts** (double _maxContacts)

  *: Remove this function, and replace it with a more generic property map*

- void **SetMaxStepSize** (double _stepSize)

  *Set max step size.*

- virtual void **SetParam** (std::string _key, const boost::any &_value)

  *Set a parameter of the physics engine.*
- void **SetRealTimeUpdateRate** (double _rate)

  *Set real time update rate.*
- virtual void **SetSeed** (uint32_t _seed)=0

  *Set the random number seed for the physics engine.*
- virtual void **SetSORPGSIters** (unsigned int _iters)

  *: Remove this function, and replace it with a more generic property map*
- virtual void **SetSORPGSPreconIters** (unsigned int _iters)

  *: Remove this function, and replace it with a more generic property map*
- virtual void **SetSORPGSW** (double _w)

  *: Remove this function, and replace it with a more generic property map*
- void **SetTargetRealTimeFactor** (double _factor)

  *Set target real time factor.*
- virtual void **SetWorldCFM** (double _cfm)

  *: Remove this function, and replace it with a more generic property map*
- virtual void **SetWorldERP** (double _erp)

  *: Remove this function, and replace it with a more generic property map*
- virtual void **UpdateCollision** ()=0

  *Update the physics engine collision.*
- virtual void **UpdatePhysics** ()

  *Update the physics engine.*

## Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)

  *virtual callback for gztopic "~/physics".*
- virtual void **OnRequest** (ConstRequestPtr &_msg)

  *virtual callback for gztopic "~/request".*

## Protected Attributes

- **ContactManager** * **contactManager**

  *Class that handles all contacts generated by the physics engine.*
- double **maxStepSize**

  *Real time update rate.*
- **transport::NodePtr node**

  *Node for communication.*
- **transport::SubscriberPtr physicsSub**

  *Subscribe to the physics topic.*
- boost::recursive_mutex * **physicsUpdateMutex**

  *Mutex to protect the update cycle.*
- double **realTimeUpdateRate**

  *Real time update rate.*
- **transport::SubscriberPtr requestSub**

  *Subscribe to the request topic.*

- **transport::PublisherPtr responsePub**

  *Response publisher.*
- sdf::ElementPtr **sdf**

  *Our SDF values.*
- double **targetRealTimeFactor**

  *Target real time factor.*
- **WorldPtr world**

  *Pointer to the world.*

### 10.132.1 Detailed Description

**Base** (p. 159) class for a physics engine.

### 10.132.2 Constructor & Destructor Documentation

#### 10.132.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine ( WorldPtr *_world* ) [explicit]

Default constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to the world. |

#### 10.132.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine ( ) [virtual]

Destructor.

### 10.132.3 Member Function Documentation

#### 10.132.3.1 virtual CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision ( const std::string & *_shapeType,* LinkPtr *_link* ) [pure virtual]

Create a collision.

**Parameters**

| | | |
|---|---|---|
| in | *_shapeType* | Type of collision to create. |
| in | *_link* | Parent link. |

Implemented in **gazebo::physics::DARTPhysics** (p. 332), and **gazebo::physics::SimbodyPhysics** (p. 918).

#### 10.132.3.2 CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision ( const std::string & *_shapeType,* const std::string & *_linkName* )

Create a collision.

**Parameters**

| | | |
|---|---|---|
| in | *_shapeType* | Type of collision to create. |
| in | *_linkName* | Name of the parent link. |

#### 10.132.3.3 virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint ( const std::string & *_type*, ModelPtr *_parent* = ModelPtr() ) [pure virtual]

Create a new joint.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | Type of joint to create. |
| in | *_parent* | **Model** (p. 624) parent. |

Implemented in **gazebo::physics::DARTPhysics** (p. 332), and **gazebo::physics::SimbodyPhysics** (p. 918).

#### 10.132.3.4 virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink ( ModelPtr *_parent* ) [pure virtual]

Create a new body.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent model for the link. |

Implemented in **gazebo::physics::DARTPhysics** (p. 332), and **gazebo::physics::SimbodyPhysics** (p. 919).

#### 10.132.3.5 virtual ModelPtr gazebo::physics::PhysicsEngine::CreateModel ( BasePtr *_base* ) [virtual]

Create a new model.

**Parameters**

| | | |
|---|---|---|
| in | *_base* | Boost shared pointer to a new model. |

Reimplemented in **gazebo::physics::DARTPhysics** (p. 333), and **gazebo::physics::SimbodyPhysics** (p. 919).

#### 10.132.3.6 virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape ( const std::string & *_shapeType*, CollisionPtr *_collision* ) [pure virtual]

Create a **physics::Shape** (p. 861) object.

**Parameters**

| | | |
|---|---|---|
| in | *_shapeType* | Type of shape to create. |
| in | *_collision* | **Collision** (p. 220) parent. |

Implemented in **gazebo::physics::DARTPhysics** (p. 333), and **gazebo::physics::SimbodyPhysics** (p. 919).

#### 10.132.3.7 virtual void gazebo::physics::PhysicsEngine::DebugPrint ( ) const [pure virtual]

Debug print out of the physic engine state.

Implemented in **gazebo::physics::DARTPhysics** (p. 333), and **gazebo::physics::SimbodyPhysics** (p. 919).

#### 10.132.3.8 virtual void gazebo::physics::PhysicsEngine::Fini ( ) [virtual]

Finilize the physics engine.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 333), and **gazebo::physics::SimbodyPhysics** (p. 919).

#### 10.132.3.9 virtual bool gazebo::physics::PhysicsEngine::GetAutoDisableFlag ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters..

**Returns**

Auto disable flag.

#### 10.132.3.10 ContactManager* gazebo::physics::PhysicsEngine::GetContactManager ( ) const

Get a pointer to the contact manger.

**Returns**

Pointer to the contact manager.

#### 10.132.3.11 virtual double gazebo::physics::PhysicsEngine::GetContactMaxCorrectingVel ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

**Returns**

Max correcting velocity.

#### 10.132.3.12 virtual double gazebo::physics::PhysicsEngine::GetContactSurfaceLayer ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

**Returns**

**Contact** (p. 260) suerface layer depth.

#### 10.132.3.13 virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity ( ) const [virtual]

Return the gavity vector.

**Returns**

The gavity vector.

#### 10.132.3.14 virtual int gazebo::physics::PhysicsEngine::GetMaxContacts ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

**Returns**

Maximum number of allows contacts.

#### 10.132.3.15 double gazebo::physics::PhysicsEngine::GetMaxStepSize ( ) const

Get max step size.

**Returns**

Max step size.

#### 10.132.3.16 virtual boost::any gazebo::physics::PhysicsEngine::GetParam ( std::string _key ) const [virtual]

Get an parameter of the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_attr* | String key |

**Returns**

The value of the parameter

#### 10.132.3.17 boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex ( ) const [inline]

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 720).

**Returns**

Pointer to the physics mutex.

References physicsUpdateMutex.

#### 10.132.3.18 double gazebo::physics::PhysicsEngine::GetRealTimeUpdateRate ( ) const

Get real time update rate.

**Returns**

Update rate

#### 10.132.3.19 virtual int gazebo::physics::PhysicsEngine::GetSORPGSIters ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters.

**Returns**

SORPGS iterations.

#### 10.132.3.20 virtual int gazebo::physics::PhysicsEngine::GetSORPGSPreconIters ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters.

**Returns**

SORPGS precondition iterations.

#### 10.132.3.21 virtual double gazebo::physics::PhysicsEngine::GetSORPGSW ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters

**Returns**

SORPGSW value.

#### 10.132.3.22 double gazebo::physics::PhysicsEngine::GetTargetRealTimeFactor ( ) const

Get target real time factor.

**Returns**

Target real time factor

#### 10.132.3.23 virtual std::string gazebo::physics::PhysicsEngine::GetType ( ) const [pure virtual]

Return the physics engine type (ode|bullet|dart|simbody).

**Returns**

Type of the physics engine.

Implemented in **gazebo::physics::DARTPhysics** (p. 334), and **gazebo::physics::SimbodyPhysics** (p. 920).

#### 10.132.3.24 double gazebo::physics::PhysicsEngine::GetUpdatePeriod ( )

Get the simulation update period.

**Returns**

Simulation update period.

#### 10.132.3.25 virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1157) CFM.

**Returns**

**World** (p. 1157) CFM.

#### 10.132.3.26 virtual double gazebo::physics::PhysicsEngine::GetWorldERP ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1157) ERP.

**Returns**

**World** (p. 1157) ERP.

#### 10.132.3.27 virtual void gazebo::physics::PhysicsEngine::Init ( ) [pure virtual]

Initialize the physics engine.

Implemented in **gazebo::physics::DARTPhysics** (p. 334), and **gazebo::physics::SimbodyPhysics** (p. 920).

#### 10.132.3.28 virtual void gazebo::physics::PhysicsEngine::InitForThread ( ) [pure virtual]

Init the engine for threads.

Implemented in **gazebo::physics::DARTPhysics** (p. 334), and **gazebo::physics::SimbodyPhysics** (p. 920).

#### 10.132.3.29 virtual void gazebo::physics::PhysicsEngine::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to the SDF parameters. |

Reimplemented in **gazebo::physics::DARTPhysics** (p. 334), and **gazebo::physics::SimbodyPhysics** (p. 921).

#### 10.132.3.30 virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg ( ConstPhysicsPtr & *_msg* ) [protected], [virtual]

virtual callback for gztopic "~/physics".

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Physics message. |

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 921), and **gazebo::physics::DARTPhysics** (p. 334).

#### 10.132.3.31 virtual void gazebo::physics::PhysicsEngine::OnRequest ( ConstRequestPtr & *_msg* ) [protected], [virtual]

virtual callback for gztopic "~/request".

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Request message. |

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 921), and **gazebo::physics::DARTPhysics** (p. 335).

#### 10.132.3.32 virtual void gazebo::physics::PhysicsEngine::Reset ( ) [inline], [virtual]

Rest the physics engine.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 335), and **gazebo::physics::SimbodyPhysics** (p. 922).

#### 10.132.3.33 virtual void gazebo::physics::PhysicsEngine::SetAutoDisableFlag ( bool *_autoDisable* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_autoDisable* | True to enable auto disabling of bodies. |

#### 10.132.3.34 virtual void gazebo::physics::PhysicsEngine::SetContactMaxCorrectingVel ( double *_vel* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Max correcting velocity. |

#### 10.132.3.35 virtual void gazebo::physics::PhysicsEngine::SetContactSurfaceLayer ( double *_layerDepth* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_layerDepth* | Surface layer depth |

#### 10.132.3.36 virtual void gazebo::physics::PhysicsEngine::SetGravity ( const gazebo::math::Vector3 & *_gravity* ) [pure virtual]

Set the gavity vector.

**Parameters**

| | | |
|---|---|---|
| in | *_gravity* | New gravity vector. |

Implemented in **gazebo::physics::DARTPhysics** (p. 335), and **gazebo::physics::SimbodyPhysics** (p. 922).

#### 10.132.3.37 virtual void gazebo::physics::PhysicsEngine::SetMaxContacts ( double *_maxContacts* ) [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters

**Parameters**

| | | |
|---|---|---|
| in | *_maxContacts* | Maximum number of contacts. |

#### 10.132.3.38 void gazebo::physics::PhysicsEngine::SetMaxStepSize ( double *_stepSize* )

Set max step size.

**Parameters**

| | | |
|---|---|---|
| in | *_stepSize* | Max step size. |

#### 10.132.3.39 virtual void gazebo::physics::PhysicsEngine::SetParam ( std::string *_key,* const boost::any & *_value* ) [virtual]

Set a parameter of the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | String key |
| in | *_value* | The value to set to |

#### 10.132.3.40 void gazebo::physics::PhysicsEngine::SetRealTimeUpdateRate ( double *_rate* )

Set real time update rate.

**Parameters**

| | | |
|---|---|---|
| in | *_rate* | Update rate |

#### 10.132.3.41 virtual void gazebo::physics::PhysicsEngine::SetSeed ( uint32_t *_seed* ) [pure virtual]

Set the random number seed for the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_seed* | The random number seed. |

Implemented in **gazebo::physics::DARTPhysics** (p. 335), and **gazebo::physics::SimbodyPhysics** (p. 922).

#### 10.132.3.42 virtual void gazebo::physics::PhysicsEngine::SetSORPGSIters ( unsigned int *_iters* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_iter* | Number of iterations. |

#### 10.132.3.43 virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconIters ( unsigned int *_iters* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_iter* | Number of iterations. |

#### 10.132.3.44 virtual void gazebo::physics::PhysicsEngine::SetSORPGSW ( double *_w* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_w* | SORPGSW value. |

#### 10.132.3.45 void gazebo::physics::PhysicsEngine::SetTargetRealTimeFactor ( double *_factor* )

Set target real time factor.

**Parameters**

| | | |
|---|---|---|
| in | *_factor* | Target real time factor |

#### 10.132.3.46 virtual void gazebo::physics::PhysicsEngine::SetWorldCFM ( double *_cfm* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_cfm* | Constraint force mixing. |

#### 10.132.3.47 virtual void gazebo::physics::PhysicsEngine::SetWorldERP ( double *_erp* ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_erp* | Error reduction parameter. |

#### 10.132.3.48 virtual void gazebo::physics::PhysicsEngine::UpdateCollision ( ) [pure virtual]

Update the physics engine collision.

Implemented in **gazebo::physics::DARTPhysics** (p. 335), and **gazebo::physics::SimbodyPhysics** (p. 923).

#### 10.132.3.49 virtual void gazebo::physics::PhysicsEngine::UpdatePhysics ( ) [inline], [virtual]

Update the physics engine.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 335), and **gazebo::physics::SimbodyPhysics** (p. 923).

### 10.132.4 Member Data Documentation

#### 10.132.4.1 ContactManager* gazebo::physics::PhysicsEngine::contactManager [protected]

Class that handles all contacts generated by the physics engine.

#### 10.132.4.2 double gazebo::physics::PhysicsEngine::maxStepSize [protected]

Real time update rate.

#### 10.132.4.3 transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]

Node for communication.

#### 10.132.4.4 transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub [protected]

Subscribe to the physics topic.

#### 10.132.4.5 boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex [protected]

Mutex to protect the update cycle.

Referenced by GetPhysicsUpdateMutex().

#### 10.132.4.6 double gazebo::physics::PhysicsEngine::realTimeUpdateRate [protected]

Real time update rate.

#### 10.132.4.7 transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub [protected]

Subscribe to the request topic.

#### 10.132.4.8 transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub [protected]

Response publisher.

#### 10.132.4.9 sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf [protected]

Our SDF values.

#### 10.132.4.10 double gazebo::physics::PhysicsEngine::targetRealTimeFactor [protected]

Target real time factor.

#### 10.132.4.11 WorldPtr gazebo::physics::PhysicsEngine::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **PhysicsEngine.hh**

## 10.133 gazebo::physics::PhysicsFactory Class Reference

The physics factory instantiates different physics engines.

`#include <physics/physics.hh>`

### Static Public Member Functions

- static bool **IsRegistered** (const std::string &_name)

*Check if a physics engine is registered.*

- static **PhysicsEnginePtr NewPhysicsEngine** (const std::string &_className, **WorldPtr** _world)

  *Create a new instance of a physics engine.*
- static void **RegisterAll** ()

  *Register everything.*
- static void **RegisterPhysicsEngine** (std::string _className, **PhysicsFactoryFn** _factoryfn)

  *Register a physics class.*

### 10.133.1 Detailed Description

The physics factory instantiates different physics engines.

### 10.133.2 Member Function Documentation

#### 10.133.2.1 static bool gazebo::physics::PhysicsFactory::IsRegistered ( const std::string & *_name* ) [static]

Check if a physics engine is registered.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the physics engine. |

**Returns**

True if physics engine is registered, false otherwise.

#### 10.133.2.2 static PhysicsEnginePtr gazebo::physics::PhysicsFactory::NewPhysicsEngine ( const std::string & *_className,* WorldPtr *_world* ) [static]

Create a new instance of a physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_className* | Name of the physics class. |
| in | *_world* | **World** (p. 1157) to pass to the created physics engine. |

#### 10.133.2.3 static void gazebo::physics::PhysicsFactory::RegisterAll ( ) [static]

Register everything.

#### 10.133.2.4 static void gazebo::physics::PhysicsFactory::RegisterPhysicsEngine ( std::string *_className,* PhysicsFactoryFn *_factoryfn* ) [static]

Register a physics class.

**Parameters**

| | | |
|---|---|---|
| in | *_className* | Name of the physics class. |
| in | *_factoryfn* | Function pointer used to create a physics engine. |

The documentation for this class was generated from the following file:

- **PhysicsFactory.hh**

## 10.134 gazebo::common::PID Class Reference

Generic **PID** (p. 722) controller class.

`#include <common/common.hh>`

### Public Member Functions

- **PID** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)

  *Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].*
- virtual ~**PID** ()

  *Destructor.*
- double **GetCmd** ()

  *Return current command for this **PID** (p. 722) controller.*
- void **GetErrors** (double &_pe, double &_ie, double &_de)

  *Return **PID** (p. 722) error terms for the controller.*
- void **Init** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)

  *Initialize PID-gains and integral term limits:[iMax:iMin]-[I1:I2].*
- **PID** & **operator=** (const **PID** &_p)

  *Assignment operator.*
- void **Reset** ()

  *Reset the errors and command.*
- void **SetCmd** (double _cmd)

  *Set current target command for this **PID** (p. 722) controller.*
- void **SetCmdMax** (double _c)

  *Set the maximum value for the command.*
- void **SetCmdMin** (double _c)

  *Set the maximum value for the command.*
- void **SetDGain** (double _d)

  *Set the derivtive Gain.*
- void **SetIGain** (double _i)

  *Set the integral Gain.*
- void **SetIMax** (double _i)

  *Set the integral upper limit.*
- void **SetIMin** (double _i)

  *Set the integral lower limit.*
- void **SetPGain** (double _p)

  *Set the proportional Gain.*
- double **Update** (double _error, **common::Time** _dt)

  *Update the Pid loop with nonuniform time step size.*

### 10.134.1 Detailed Description

Generic **PID** (p. 722) controller class.

Generic proportiolnal-integral-derivative controller class that keeps track of PID-error states and control inputs given the state of a system and a user specified target state.

### 10.134.2 Constructor & Destructor Documentation

#### 10.134.2.1 gazebo::common::PID::PID ( double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0 )

Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].

**Parameters**

| | | |
|---|---|---|
| in | *_p* | The proportional gain. |
| in | *_i* | The integral gain. |
| in | *_d* | The derivative gain. |
| in | *_imax* | The integral upper limit. |
| in | *_imin* | The integral lower limit. |
| in | *_cmdMax* | Output max value. |
| in | *_cmdMin* | Output min value. |

#### 10.134.2.2 virtual gazebo::common::PID::~PID ( ) [virtual]

Destructor.

### 10.134.3 Member Function Documentation

#### 10.134.3.1 double gazebo::common::PID::GetCmd ( )

Return current command for this **PID** (p. 722) controller.

**Returns**

the command value

#### 10.134.3.2 void gazebo::common::PID::GetErrors ( double & _pe, double & _ie, double & _de )

Return **PID** (p. 722) error terms for the controller.

**Parameters**

| | | |
|---|---|---|
| in | *_pe* | The proportional error. |
| in | *_ie* | The integral error. |
| in | *_de* | The derivative error. |

#### 10.134.3.3 void gazebo::common::PID::Init ( double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0 )

Initialize PID-gains and integral term limits:[iMax:iMin]-[I1:I2].

**Parameters**

| | | |
|---|---|---|
| in | *_p* | The proportional gain. |
| in | *_i* | The integral gain. |
| in | *_d* | The derivative gain. |
| in | *_imax* | The integral upper limit. |
| in | *_imin* | The integral lower limit. |
| in | *_cmdMax* | Output max value. |
| in | *_cmdMin* | Output min value. |

#### 10.134.3.4 PID& gazebo::common::PID::operator= ( const PID & _p ) [inline]

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_p* | a reference to a **PID** (p. 722) to assign values from |

**Returns**

reference to this instance

References Reset().

#### 10.134.3.5 void gazebo::common::PID::Reset ( )

Reset the errors and command.

Referenced by operator=().

#### 10.134.3.6 void gazebo::common::PID::SetCmd ( double _cmd )

Set current target command for this **PID** (p. 722) controller.

**Parameters**

| | | |
|---|---|---|
| in | *_cmd* | New command |

#### 10.134.3.7 void gazebo::common::PID::SetCmdMax ( double _c )

Set the maximum value for the command.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The maximum value |

#### 10.134.3.8 void gazebo::common::PID::SetCmdMin ( double _c )

Set the maximum value for the command.

**Parameters**

| | | |
|---|---|---|
| in | _c | The maximum value |

#### 10.134.3.9 void gazebo::common::PID::SetDGain ( double _d )

Set the derivtive Gain.

**Parameters**

| | | |
|---|---|---|
| in | _p | dertivative gain value |

#### 10.134.3.10 void gazebo::common::PID::SetIGain ( double _i )

Set the integral Gain.

**Parameters**

| | | |
|---|---|---|
| in | _p | integral gain value |

#### 10.134.3.11 void gazebo::common::PID::SetIMax ( double _i )

Set the integral upper limit.

**Parameters**

| | | |
|---|---|---|
| in | _p | integral upper limit value |

#### 10.134.3.12 void gazebo::common::PID::SetIMin ( double _i )

Set the integral lower limit.

**Parameters**

| | | |
|---|---|---|
| in | _p | integral lower limit value |

#### 10.134.3.13 void gazebo::common::PID::SetPGain ( double _p )

Set the proportional Gain.

**Parameters**

| | | |
|---|---|---|
| in | _p | proportional gain value |

#### 10.134.3.14 double gazebo::common::PID::Update ( double *_error,* common::Time *_dt* )

Update the Pid loop with nonuniform time step size.

**Parameters**

| | |
|---|---|
| *_in]* | _error Error since last call (p_state - p_target). |
| *_in]* | _dt Change in time since last update call. Normally, this is called at every time step, The return value is an updated command to be passed to the object being controlled. |

**Returns**

the command value

The documentation for this class was generated from the following file:

- **PID.hh**

## 10.135 gazebo::math::Plane Class Reference

A plane and related functions.

`#include <math/gzmath.hh>`

### Public Member Functions

- **Plane** ()
  *Constructor.*
- **Plane** (const **Vector3** &_normal, double _offset=0.0)
  *Constructor from a normal and a distanec.*
- **Plane** (const **Vector3** &_normal, const **Vector2d** &_size, double _offset)
  *Constructor.*
- virtual ~**Plane** ()
  *Destructor.*
- double **Distance** (const **Vector3** &_origin, const **Vector3** &_dir) const
  *Get distance to the plane give an origin and direction.*
- **Plane** & **operator=** (const **Plane** &_p)
  *Equal operator.*
- void **Set** (const **Vector3** &_normal, const **Vector2d** &_size, double offset)
  *Set the plane.*

### Public Attributes

- double **d**
  ***Plane*** *(*p. *726) offset.*
- **Vector3 normal**
  ***Plane*** *(*p. *726) normal.*
- **Vector2d size**
  ***Plane*** *(*p. *726) size.*

### 10.135.1 Detailed Description

A plane and related functions.

### 10.135.2 Constructor & Destructor Documentation

#### 10.135.2.1 gazebo::math::Plane::Plane ( )

Constructor.

#### 10.135.2.2 gazebo::math::Plane::Plane ( const Vector3 & *_normal,* double *_offset* = 0.0 )

Constructor from a normal and a distanec.

**Parameters**

| | | |
|---|---|---|
| in | *_normal* | The plane normal |
| in | *_offset* | Offset along the normal |

#### 10.135.2.3 gazebo::math::Plane::Plane ( const Vector3 & *_normal,* const Vector2d & *_size,* double *_offset* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_normal* | The plane normal |
| in | *_size* | Size of the plane |
| in | *_offset* | Offset along the normal |

#### 10.135.2.4 virtual gazebo::math::Plane::~Plane ( ) [virtual]

Destructor.

### 10.135.3 Member Function Documentation

#### 10.135.3.1 double gazebo::math::Plane::Distance ( const Vector3 & *_origin,* const Vector3 & *_dir* ) const

Get distance to the plane give an origin and direction.

**Parameters**

| | | |
|---|---|---|
| in | *_origin* | the origin |
| in | *_dir* | a direction |

**Returns**

the shortest distance

#### 10.135.3.2 Plane& gazebo::math::Plane::operator= ( const Plane & _p )

Equal operator.

**Parameters**

| | |
|---|---|
| *_p* | another plane |

**Returns**

itself

#### 10.135.3.3 void gazebo::math::Plane::Set ( const Vector3 & _normal, const Vector2d & _size, double offset )

Set the plane.

**Parameters**

| | | |
|---|---|---|
| in | *_normal* | The plane normal |
| in | *_size* | Size of the plane |
| in | *_offset* | Offset along the normal |

### 10.135.4 Member Data Documentation

#### 10.135.4.1 double gazebo::math::Plane::d

**Plane** (p. 726) offset.

#### 10.135.4.2 Vector3 gazebo::math::Plane::normal

**Plane** (p. 726) normal.

#### 10.135.4.3 Vector2d gazebo::math::Plane::size

**Plane** (p. 726) size.

The documentation for this class was generated from the following file:

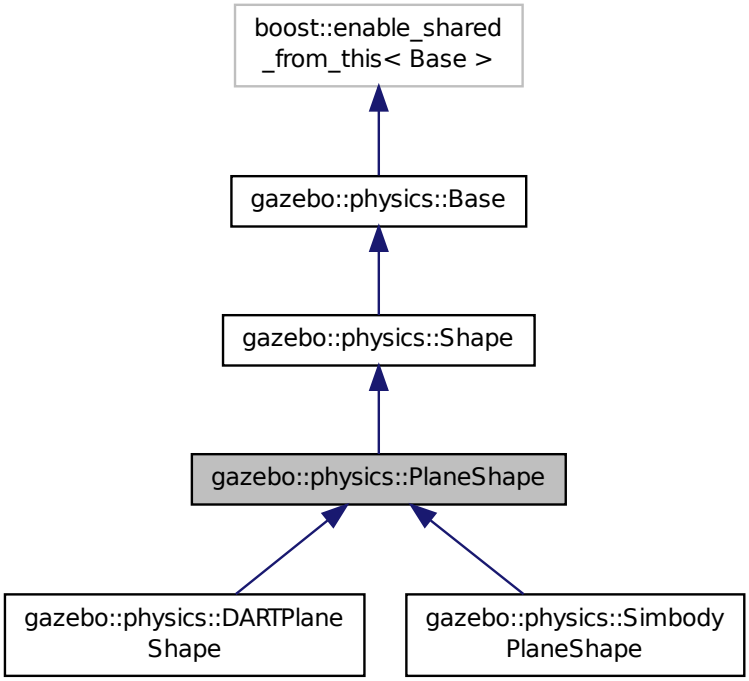
- **Plane.hh**

## 10.136 gazebo::physics::PlaneShape Class Reference

**Collision** (p. 220) for an infinite plane.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::PlaneShape:




## Public Member Functions

- **PlaneShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**PlaneShape** ()

  *Destructor.*
- virtual void **CreatePlane** ()

  *Create the plane.*
- void **FillMsg** (msgs::Geometry &_msg)

  *Fill a geometry message with data from this object.*
- **math::Vector3 GetNormal** () const

  *Get the plane normal.*
- **math::Vector2d GetSize** () const

  *Get the size.*
- virtual void **Init** ()

  *Initialize the plane.*
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *Process a geometry message and use the data to update this object.*
- virtual void **SetAltitude** (const **math::Vector3** &_pos)

*Set the altitude of the plane.*

- void **SetNormal** (const **math::Vector3** &_norm)

  *Set the normal.*

- virtual void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the plane.*

- void **SetSize** (const **math::Vector2d** &_size)

  *Set the size.*

### Additional Inherited Members

### 10.136.1 Detailed Description

**Collision** (p. 220) for an infinite plane.

This collision is used primarily for ground planes. Note that while the plane in infinite, only the part near the camera is drawn.

### 10.136.2 Constructor & Destructor Documentation

#### 10.136.2.1 gazebo::physics::PlaneShape::PlaneShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | **Link** (p. 542) to which we are attached. |

#### 10.136.2.2 virtual gazebo::physics::PlaneShape::~PlaneShape ( ) [virtual]

Destructor.

### 10.136.3 Member Function Documentation

#### 10.136.3.1 virtual void gazebo::physics::PlaneShape::CreatePlane ( ) [virtual]

Create the plane.

Reimplemented in **gazebo::physics::SimbodyPlaneShape** (p. 926), and **gazebo::physics::DARTPlaneShape** (p. 337).

Referenced by gazebo::physics::DARTPlaneShape::CreatePlane().

#### 10.136.3.2 void gazebo::physics::PlaneShape::FillMsg ( msgs::Geometry & *_msg* ) [virtual]

Fill a geometry message with data from this object.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message to fill. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.136.3.3 math::Vector3 gazebo::physics::PlaneShape::GetNormal ( ) const

Get the plane normal.

**Returns**

The plane normal.

#### 10.136.3.4 math::Vector2d gazebo::physics::PlaneShape::GetSize ( ) const

Get the size.

**Returns**

Size of the plane.

#### 10.136.3.5 virtual void gazebo::physics::PlaneShape::Init ( ) [virtual]

Initialize the plane.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.136.3.6 virtual void gazebo::physics::PlaneShape::ProcessMsg ( const msgs::Geometry & *_msg* ) [virtual]

Process a geometry message and use the data to update this object.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to update from. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.136.3.7 virtual void gazebo::physics::PlaneShape::SetAltitude ( const math::Vector3 & *_pos* ) [virtual]

Set the altitude of the plane.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | Position of the plane. |

Reimplemented in **gazebo::physics::DARTPlaneShape** (p. 337), and **gazebo::physics::SimbodyPlaneShape** (p. 926).

Referenced by gazebo::physics::DARTPlaneShape::SetAltitude().

#### 10.136.3.8 void gazebo::physics::PlaneShape::SetNormal ( const math::Vector3 & *_norm* )

Set the normal.

**Parameters**

| | | |
|---|---|---|
| in | *_norm* | Plane normal. |

#### 10.136.3.9 virtual void gazebo::physics::PlaneShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set the scale of the plane.

**Returns**

_scale Scale to set the plane to.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.136.3.10 void gazebo::physics::PlaneShape::SetSize ( const math::Vector2d & *_size* )

Set the size.

**Parameters**

| | | |
|---|---|---|
| in | *_size* | 2D size of the plane. |

The documentation for this class was generated from the following file:

- **PlaneShape.hh**

## 10.137 gazebo::PluginT< T > Class Template Reference

A class which all plugins must inherit from.

`#include <common/common.hh>`

### Public Types

- typedef boost::shared_ptr< T > **TPtr**
  *plugin pointer type definition*

### Public Member Functions

- **PluginT** ()
  *Constructor.*
- virtual ~**PluginT** ()
  *Destructor.*
- std::string **GetFilename** () const
  *Get the name of the handler.*
- std::string **GetHandle** () const
  *Get the short name of the handler.*
- **PluginType GetType** () const
  *Returns the type of the plugin.*

### Static Public Member Functions

- static **TPtr Create** (const std::string &_filename, const std::string &_handle)

  *a class method that creates a plugin from a file name.*

### Protected Attributes

- std::string **filename**

  *Path to the shared library file.*
- std::string **handle**

  *Short name.*
- **PluginType type**

  *Type of plugin.*

## 10.137.1 Detailed Description

**template<class T>class gazebo::PluginT< T >**

A class which all plugins must inherit from.

## 10.137.2 Member Typedef Documentation

### 10.137.2.1 template<class T> typedef boost::shared_ptr<T> gazebo::PluginT< T >::TPtr

plugin pointer type definition

## 10.137.3 Constructor & Destructor Documentation

### 10.137.3.1 template<class T> gazebo::PluginT< T >::PluginT ( ) [inline]

Constructor.

### 10.137.3.2 template<class T> virtual gazebo::PluginT< T >::~PluginT ( ) [inline], [virtual]

Destructor.

## 10.137.4 Member Function Documentation

### 10.137.4.1 template<class T> static TPtr gazebo::PluginT< T >::Create ( const std::string & *_filename*, const std::string & *_handle* ) [inline], [static]

a class method that creates a plugin from a file name.

It locates the shared library and loads it dynamically.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | the path to the shared library. |
| in | *_handle* | short name of the handler |

**Returns**

Shared Pointer to this class type

#### 10.137.4.2 template<class T> std::string gazebo::PluginT< T >::GetFilename ( ) const [inline]

Get the name of the handler.

#### 10.137.4.3 template<class T> std::string gazebo::PluginT< T >::GetHandle ( ) const [inline]

Get the short name of the handler.

#### 10.137.4.4 template<class T> PluginType gazebo::PluginT< T >::GetType ( ) const [inline]

Returns the type of the plugin.

**Returns**

type of the plugin

### 10.137.5 Member Data Documentation

#### 10.137.5.1 template<class T> std::string gazebo::PluginT< T >::filename [protected]

Path to the shared library file.

Referenced by gazebo::PluginT< ModelPlugin >::GetFilename().

#### 10.137.5.2 template<class T> std::string gazebo::PluginT< T >::handle [protected]

Short name.

Referenced by gazebo::PluginT< ModelPlugin >::Create(), and gazebo::PluginT< ModelPlugin >::GetHandle().

#### 10.137.5.3 template<class T> PluginType gazebo::PluginT< T >::type [protected]

Type of plugin.

Referenced by gazebo::PluginT< ModelPlugin >::GetType().

The documentation for this class was generated from the following file:

- **Plugin.hh**

## 10.138 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

`#include <math/gzmath.hh>`

## Public Member Functions

- **Pose** ()

  *Default constructors.*
- **Pose** (const **Vector3** &_pos, const **Quaternion** &_rot)

  *Constructor.*
- **Pose** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

  *Constructor.*
- **Pose** (const **Pose** &_pose)

  *Copy constructor.*
- virtual ∼**Pose** ()

  *Destructor.*
- **Pose CoordPoseSolve** (const **Pose** &_b) const

  *Find the inverse of a pose; i.e., if b = this + a, given b and this, find a.*
- **Vector3 CoordPositionAdd** (const **Vector3** &_pos) const

  *Add one point to a vector: result = this + pos.*
- **Vector3 CoordPositionAdd** (const **Pose** &_pose) const

  *Add one point to another: result = this + pose.*
- **Vector3 CoordPositionSub** (const **Pose** &_pose) const

  *Subtract one position from another: result = this - pose.*
- **Quaternion CoordRotationAdd** (const **Quaternion** &_rot) const

  *Add one rotation to another: result = this->rot + rot.*
- **Quaternion CoordRotationSub** (const **Quaternion** &_rot) const

  *Subtract one rotation from another: result = this->rot - rot.*
- void **Correct** ()

  *Fix any nan values.*
- **Pose GetInverse** () const

  *Get the inverse of this pose.*
- bool **IsFinite** () const

  *See if a pose is finite (e.g., not nan)*
- bool **operator!=** (const **Pose** &_pose) const

  *Inequality operator.*
- **Pose operator**∗ (const **Pose** &_pose)

  *Multiplication operator.*
- **Pose operator+** (const **Pose** &_pose) const

  *Addition operator A is the transform from O to P specified in frame O B is the transform from P to Q specified in frame P then, B + A is the transform from O to Q specified in frame O.*
- const **Pose** & **operator+=** (const **Pose** &_pose)

  *Add-Equals operator.*
- **Pose operator-** () const

  *Negation operator A is the transform from O to P in frame O then -A is transform from P to O specified in frame P.*
- **Pose operator-** (const **Pose** &_pose) const

  *Subtraction operator A is the transform from O to P in frame O B is the transform from O to Q in frame O B - A is the transform from P to Q in frame P.*
- const **Pose** & **operator-=** (const **Pose** &_pose)

  *Subtraction operator.*
- **Pose** & **operator=** (const **Pose** &_pose)

  *Equal operator.*

- bool **operator==** (const **Pose** &_pose) const

  *Equality operator.*
- void **Reset** ()

  *Reset the pose.*
- **Pose RotatePositionAboutOrigin** (const **Quaternion** &_rot) const

  *Rotate vector part of a pose about the origin.*
- void **Round** (int _precision)

  *Round all values to _precision decimal places.*
- void **Set** (const **Vector3** &_pos, const **Quaternion** &_rot)

  *Set the pose from a **Vector3*** (p. *1091) and a **Quaternion*** (p. *761).*
- void **Set** (const **Vector3** &_pos, const **Vector3** &_rpy)

  *Set the pose from pos and rpy vectors.*
- void **Set** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

  *Set the pose from a six tuple.*

### Public Attributes

- **Vector3 pos**

  *The position.*
- **Quaternion rot**

  *The rotation.*

### Static Public Attributes

- static const **Pose Zero**

  *math::Pose(0, 0, 0, 0, 0, 0)*

### Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Pose** &_pose)

  *Stream insertion operator.*
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Pose** &_pose)

  *Stream extraction operator.*

## 10.138.1 Detailed Description

Encapsulates a position and rotation in three space.

## 10.138.2 Constructor & Destructor Documentation

### 10.138.2.1 gazebo::math::Pose::Pose ( )

Default constructors.

Referenced by operator-().

#### 10.138.2.2 gazebo::math::Pose::Pose ( const Vector3 & _pos, const Quaternion & _rot )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | A position |
| in | *_rot* | A rotation |

#### 10.138.2.3 gazebo::math::Pose::Pose ( double _x, double _y, double _z, double _roll, double _pitch, double _yaw )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | x position in meters. |
| in | *_y* | y position in meters. |
| in | *_z* | z position in meters. |
| in | *_roll* | Roll (rotation about X-axis) in radians. |
| in | *_pitch* | Pitch (rotation about y-axis) in radians. |
| in | *_yaw* | Yaw (rotation about z-axis) in radians. |

#### 10.138.2.4 gazebo::math::Pose::Pose ( const Pose & _pose )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) to copy |

#### 10.138.2.5 virtual gazebo::math::Pose::~Pose ( ) [virtual]

Destructor.

### 10.138.3 Member Function Documentation

#### 10.138.3.1 Pose gazebo::math::Pose::CoordPoseSolve ( const Pose & _b ) const

Find the inverse of a pose; i.e., if b = this + a, given b and this, find a.

**Parameters**

| | | |
|---|---|---|
| in | *_b* | the other pose |

#### 10.138.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd ( const Vector3 & _pos ) const

Add one point to a vector: result = this + pos.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | Position to add to this pose |

**Returns**

the resulting position

#### 10.138.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd ( const Pose & *_pose* ) const

Add one point to another: result = this + pose.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | The **Pose** (p. 734) to add |

**Returns**

The resulting position

#### 10.138.3.4 Vector3 gazebo::math::Pose::CoordPositionSub ( const Pose & *_pose* ) const [inline]

Subtract one position from another: result = this - pose.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) to subtract |

**Returns**

The resulting position

References gazebo::math::Quaternion::GetInverse(), pos, rot, gazebo::math::Vector3::x, gazebo::math::Quaternion::x, gazebo::math::Vector3::y, gazebo::math::Quaternion::y, gazebo::math::Vector3::z, and gazebo::math::Quaternion::z.

Referenced by operator-().

#### 10.138.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd ( const Quaternion & *_rot* ) const

Add one rotation to another: result = this->rot + rot.

**Parameters**

| | | |
|---|---|---|
| in | *_rot* | Rotation to add |

**Returns**

The resulting rotation

#### 10.138.3.6 Quaternion gazebo::math::Pose::CoordRotationSub ( const Quaternion & *_rot* ) const [inline]

Subtract one rotation from another: result = this->rot - rot.

**Parameters**

| | | |
|---|---|---|
| in | *_rot* | The rotation to subtract |

**Returns**

The resulting rotation

References gazebo::math::Quaternion::GetInverse(), gazebo::math::Quaternion::Normalize(), and rot.

Referenced by operator-().

#### 10.138.3.7 void gazebo::math::Pose::Correct ( ) [inline]

Fix any nan values.

References gazebo::math::Vector3::Correct(), gazebo::math::Quaternion::Correct(), pos, and rot.

#### 10.138.3.8 Pose gazebo::math::Pose::GetInverse ( ) const

Get the inverse of this pose.

**Returns**

the inverse pose

#### 10.138.3.9 bool gazebo::math::Pose::IsFinite ( ) const

See if a pose is finite (e.g., not nan)

#### 10.138.3.10 bool gazebo::math::Pose::operator!= ( const Pose & *_pose* ) const

Inequality operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) for comparison |

**Returns**

True if not equal

#### 10.138.3.11 Pose gazebo::math::Pose::operator* ( const Pose & *_pose* )

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | the other pose |

**Returns**

itself

#### 10.138.3.12 Pose gazebo::math::Pose::operator+ ( const Pose & _pose ) const

Addition operator A is the transform from O to P specified in frame O B is the transform from P to Q specified in frame P then, B + A is the transform from O to Q specified in frame O.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) to add to this pose |

**Returns**

The resulting pose

#### 10.138.3.13 const Pose& gazebo::math::Pose::operator+= ( const Pose & _pose )

Add-Equals operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) to add to this pose |

**Returns**

The resulting pose

#### 10.138.3.14 Pose gazebo::math::Pose::operator- ( ) const [inline]

Negation operator A is the transform from O to P in frame O then -A is transform from P to O specified in frame P.

**Returns**

The resulting pose

References Pose().

#### 10.138.3.15 Pose gazebo::math::Pose::operator- ( const Pose & _pose ) const [inline]

Subtraction operator A is the transform from O to P in frame O B is the transform from O to Q in frame O B - A is the transform from P to Q in frame P.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) to subtract from this one |

**Returns**

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

#### 10.138.3.16 const Pose& gazebo::math::Pose::operator-= ( const Pose & *_pose* )

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) to subtract from this one |

**Returns**

The resulting pose

#### 10.138.3.17 Pose& gazebo::math::Pose::operator= ( const Pose & *_pose* )

Equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) to copy |

#### 10.138.3.18 bool gazebo::math::Pose::operator== ( const Pose & *_pose* ) const

Equality operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | **Pose** (p. 734) for comparison |

**Returns**

True if equal

#### 10.138.3.19 void gazebo::math::Pose::Reset ( )

Reset the pose.

#### 10.138.3.20 Pose gazebo::math::Pose::RotatePositionAboutOrigin ( const Quaternion & *_rot* ) const

Rotate vector part of a pose about the origin.

**Parameters**

| | | |
|---|---|---|
| in | *_rot* | rotation |

**Returns**

the rotated pose

#### 10.138.3.21 void gazebo::math::Pose::Round ( int *_precision* )

Round all values to _precision decimal places.

**Parameters**

| | | |
|---|---|---|
| in | *_precision* | |

#### 10.138.3.22 void gazebo::math::Pose::Set ( const **Vector3** & *_pos*, const **Quaternion** & *_rot* )

Set the pose from a **Vector3** (p. 1091) and a **Quaternion** (p. 761).

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | The position. |
| in | *_rot* | The rotation. |

#### 10.138.3.23 void gazebo::math::Pose::Set ( const **Vector3** & *_pos*, const **Vector3** & *_rpy* )

Set the pose from pos and rpy vectors.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | The position. |
| in | *_rpy* | The rotation expressed as Euler angles. |

#### 10.138.3.24 void gazebo::math::Pose::Set ( double *_x*, double *_y*, double *_z*, double *_roll*, double *_pitch*, double *_yaw* )

Set the pose from a six tuple.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | x position in meters. |
| in | *_y* | y position in meters. |
| in | *_z* | z position in meters. |
| in | *_roll* | Roll (rotation about X-axis) in radians. |
| in | *_pitch* | Pitch (rotation about y-axis) in radians. |
| in | *_yaw* | Pitch (rotation about z-axis) in radians. |

### 10.138.4 Friends And Related Function Documentation

#### 10.138.4.1 std::ostream& operator<< ( std::ostream & *_out*, const **gazebo::math::Pose** & *_pose* ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_pose* | pose to output |

**Returns**

the stream

#### 10.138.4.2 std::istream& operator>> ( std::istream & *_in*, gazebo::math::Pose & *_pose* ) [friend]

Stream extraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_in* | the input stream |
| in | *_pose* | the pose |

**Returns**

the stream

### 10.138.5 Member Data Documentation

#### 10.138.5.1 Vector3 gazebo::math::Pose::pos

The position.

Referenced by gazebo::physics::DARTTypes::ConvPose(), CoordPositionSub(), Correct(), and gazebo::physics::Inertial-::GetCoG().

#### 10.138.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by gazebo::physics::DARTTypes::ConvPose(), CoordPositionSub(), CoordRotationSub(), Correct(), and operator-().

#### 10.138.5.3 const Pose gazebo::math::Pose::Zero [static]

math::Pose(0, 0, 0, 0, 0, 0)

The documentation for this class was generated from the following file:

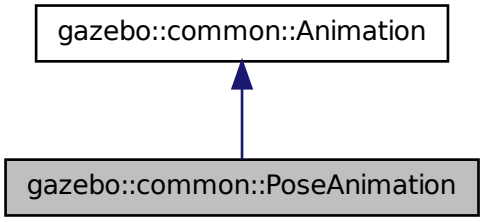
- **Pose.hh**

## 10.139 gazebo::common::PoseAnimation Class Reference

A pose animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::PoseAnimation:




### Public Member Functions

- **PoseAnimation** (const std::string &_name, double _length, bool _loop)

  *Constructor.*
- virtual ~**PoseAnimation** ()

  *Destructor.*
- **PoseKeyFrame** * **CreateKeyFrame** (double _time)

  *Create a pose keyframe at the given time.*
- void **GetInterpolatedKeyFrame** (**PoseKeyFrame** &_kf) const

  *Get a keyframe using the animation's current time.*

### Protected Member Functions

- void **BuildInterpolationSplines** () const

  *Update the pose splines.*
- void **GetInterpolatedKeyFrame** (double _time, **PoseKeyFrame** &_kf) const

  *Get a keyframe using a passed in time.*

### Additional Inherited Members

## 10.139.1 Detailed Description

A pose animation.

## 10.139.2 Constructor & Destructor Documentation

#### 10.139.2.1 gazebo::common::PoseAnimation::PoseAnimation ( const std::string & *_name,* double *_length,* bool *_loop* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | String name of the animation. This should be unique. |
| in | *_length* | Length of the animation in seconds |
| in | *_loop* | True == loop the animation |

#### 10.139.2.2 virtual gazebo::common::PoseAnimation::~PoseAnimation ( ) [virtual]

Destructor.

### 10.139.3 Member Function Documentation

#### 10.139.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines ( ) const [protected]

Update the pose splines.

#### 10.139.3.2 PoseKeyFrame* gazebo::common::PoseAnimation::CreateKeyFrame ( double *_time* )

Create a pose keyframe at the given time.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | **Time** (p. 1031) at which to create the keyframe |

**Returns**

Pointer to the new keyframe

#### 10.139.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame ( PoseKeyFrame & *_kf* ) const

Get a keyframe using the animation's current time.

**Parameters**

| | | |
|---|---|---|
| out | *_kf* | **PoseKeyFrame** (p. 746) reference to hold the interpolated result |

#### 10.139.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame ( double *_time,* PoseKeyFrame & *_kf* ) const [protected]

Get a keyframe using a passed in time.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | **Time** (p. 1031) in seconds |
| out | *_kf* | **PoseKeyFrame** (p. 746) reference to hold the interpolated result |

The documentation for this class was generated from the following file:

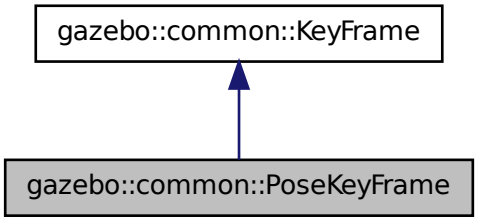
- **Animation.hh**

## 10.140 gazebo::common::PoseKeyFrame Class Reference

A keyframe for a **PoseAnimation** (p. 743).

`#include <KeyFrame.hh>`

Inheritance diagram for gazebo::common::PoseKeyFrame:




### Public Member Functions

- **PoseKeyFrame** (double _time)

  *Constructor.*
- virtual ~**PoseKeyFrame** ()

  *Destructor.*
- const **math::Quaternion** & **GetRotation** () const

  *Get the rotation of the keyframe.*
- const **math::Vector3** & **GetTranslation** () const

  *Get the translation of the keyframe.*
- void **SetRotation** (const **math::Quaternion** &_rot)

  *Set the rotation for the keyframe.*
- void **SetTranslation** (const **math::Vector3** &_trans)

  *Set the translation for the keyframe.*

### Protected Attributes

- **math::Quaternion rotate**

  *the rotation quaternion*
- **math::Vector3 translate**

  *the translation vector*

### 10.140.1 Detailed Description

A keyframe for a **PoseAnimation** (p. 743).

## 10.140.2 Constructor & Destructor Documentation

### 10.140.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame ( double *_time* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | of the keyframe |

### 10.140.2.2 virtual gazebo::common::PoseKeyFrame::~PoseKeyFrame ( ) [virtual]

Destructor.

## 10.140.3 Member Function Documentation

### 10.140.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation ( ) const

Get the rotation of the keyframe.

**Returns**

The rotation amount

### 10.140.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation ( ) const

Get the translation of the keyframe.

**Returns**

The translation amount

### 10.140.3.3 void gazebo::common::PoseKeyFrame::SetRotation ( const math::Quaternion & *_rot* )

Set the rotation for the keyframe.

**Parameters**

| | | |
|---|---|---|
| in | *_rot* | Rotation amount |

### 10.140.3.4 void gazebo::common::PoseKeyFrame::SetTranslation ( const math::Vector3 & *_trans* )

Set the translation for the keyframe.

**Parameters**

| | | |
|---|---|---|
| in | *_trans* | Translation amount |

### 10.140.4 Member Data Documentation

#### 10.140.4.1 math::Quaternion gazebo::common::PoseKeyFrame::rotate [protected]

the rotation quaternion

#### 10.140.4.2 math::Vector3 gazebo::common::PoseKeyFrame::translate [protected]

the translation vector

The documentation for this class was generated from the following file:

- **KeyFrame.hh**

## 10.141 gazebo::rendering::Projector Class Reference

Projects a material onto surface, light a light projector.

`#include <rendering/rendering.hh>`

### Public Member Functions

- **Projector** (**VisualPtr** _parent)

  *Constructor.*
- virtual ~**Projector** ()

  *Destructor.*
- **VisualPtr GetParent** ()

  *Get the parent visual.*
- void **Load** (sdf::ElementPtr _sdf)

  *Load from an sdf pointer.*
- void **Load** (const msgs::Projector &_msg)

  *Load from a message.*
- void **Load** (const std::string &_name, const **math::Pose** &_pose=**math::Pose**(0, 0, 0, 0, 0, 0), const std::string &_textureName="", double _nearClip=0.25, double _farClip=15.0, double _fov=M_PI ∗0.25)

  *Load the projector.*
- void **SetEnabled** (bool _enabled)

  *Set whether the projector is enabled or disabled.*
- void **SetTexture** (const std::string &_textureName)

  *Load a texture into the projector.*
- void **Toggle** ()

  *Toggle the activation of the projector.*

### 10.141.1 Detailed Description

Projects a material onto surface, light a light projector.

## 10.141.2 Constructor & Destructor Documentation

### 10.141.2.1 gazebo::rendering::Projector::Projector ( VisualPtr _parent )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Name of the parent visual. |

### 10.141.2.2 virtual gazebo::rendering::Projector::~Projector ( ) [virtual]

Destructor.

## 10.141.3 Member Function Documentation

### 10.141.3.1 VisualPtr gazebo::rendering::Projector::GetParent ( )

Get the parent visual.

**Returns**

Pointer ot the parent visual.

### 10.141.3.2 void gazebo::rendering::Projector::Load ( sdf::ElementPtr _sdf )

Load from an sdf pointer.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to the SDF element. |

### 10.141.3.3 void gazebo::rendering::Projector::Load ( const msgs::Projector & _msg )

Load from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Load from a message. |

### 10.141.3.4 void gazebo::rendering::Projector::Load ( const std::string & _name, const math::Pose & _pose = math::Pose(0, 0, 0, 0, 0, 0), const std::string & _textureName = " ", double _nearClip = 0.25, double _farClip = 15.0, double _fov = M_PI *0.25 )

Load the projector.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the projector. |
| in | *_pos* | Pose of the projector. |
| in | *_textureName* | Name of the texture to project. |
| in | *_nearClip* | Near clip distance. |
| in | *_farClip* | Far clip distance. |
| in | *_fov* | Field of view. |

#### 10.141.3.5 void gazebo::rendering::Projector::SetEnabled ( bool *_enabled* )

Set whether the projector is enabled or disabled.

**Parameters**

| | | |
|---|---|---|
| in | *_enabled* | True to enable the projector. |

#### 10.141.3.6 void gazebo::rendering::Projector::SetTexture ( const std::string & *_textureName* )

Load a texture into the projector.

**Parameters**

| | | |
|---|---|---|
| in | *_textureName* | Name of the texture to project. |

#### 10.141.3.7 void gazebo::rendering::Projector::Toggle ( )

Toggle the activation of the projector.

The documentation for this class was generated from the following file:

- **Projector.hh**

## 10.142 gazebo::transport::Publication Class Reference

A publication for a topic.

`#include <transport/transport.hh>`

### Public Member Functions

- **Publication** (const std::string &_topic, const std::string &_msgType)

  *Constructor.*
- virtual ~**Publication** ()

  *Destructor.*
- void **AddPublisher** (**PublisherPtr** _pub)

  *Add a publisher.*
- void **AddSubscription** (const **CallbackHelperPtr** _callback)

*Subscribe a callback to our topic.*

- void **AddSubscription** (const **NodePtr** &_node)

  *Subscribe a node to our topic.*
- void **AddTransport** (const **PublicationTransportPtr** &_publink)

  *Add a transport.*
- unsigned int **GetCallbackCount** () const

  *Get the number of callbacks.*
- bool **GetLocallyAdvertised** () const

  *Was the topic has been advertised from this process?*
- std::string **GetMsgType** () const

  *Get the type of message.*
- unsigned int **GetNodeCount** () const

  *Get the number of nodes.*
- unsigned int **GetRemoteSubscriptionCount** ()

  *Get the number of remote subscriptions.*
- unsigned int **GetTransportCount** () const

  *Get the number of transports.*
- bool **HasTransport** (const std::string &_host, unsigned int _port)

  *Does a given transport exist?*
- void **LocalPublish** (const std::string &_data)

  *Publish data to local subscribers (skip serialization)*
- void **Publish** (**MessagePtr** _msg, boost::function< void(uint32_t)> _cb, uint32_t _id)

  *Publish data to remote subscribers.*
- void **RemoveSubscription** (const **NodePtr** &_node)

  *Unsubscribe a node from our topic.*
- void **RemoveSubscription** (const std::string &_host, unsigned int _port)

  *Unsubscribe a a node by host/port from our topic.*
- void **RemoveTransport** (const std::string &_host, unsigned int _port)

  *Remove a transport.*
- void **SetLocallyAdvertised** (bool _value)

  *Set whether this topic has been advertised from this process.*

## 10.142.1 Detailed Description

A publication for a topic.

This facilitates transport of messages

## 10.142.2 Constructor & Destructor Documentation

### 10.142.2.1 gazebo::transport::Publication::Publication ( const std::string & *_topic*, const std::string & *_msgType* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic we're publishing |
| in | *_msgType* | The type of the topic we're publishing |

#### 10.142.2.2 virtual gazebo::transport::Publication::~Publication ( ) [virtual]

Destructor.

### 10.142.3 Member Function Documentation

#### 10.142.3.1 void gazebo::transport::Publication::AddPublisher ( PublisherPtr _pub )

Add a publisher.

**Parameters**

| | | |
|---|---|---|
| in,out | *_pub* | Pointer to publisher object to be added |

#### 10.142.3.2 void gazebo::transport::Publication::AddSubscription ( const CallbackHelperPtr _callback )

Subscribe a callback to our topic.

**Parameters**

| | | |
|---|---|---|
| in | *_callback* | The callback |

#### 10.142.3.3 void gazebo::transport::Publication::AddSubscription ( const NodePtr & _node )

Subscribe a node to our topic.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | The node |

#### 10.142.3.4 void gazebo::transport::Publication::AddTransport ( const PublicationTransportPtr & _publink )

Add a transport.

**Parameters**

| | | |
|---|---|---|
| in | *_publink* | Pointer to publication transport object to be added |

#### 10.142.3.5 unsigned int gazebo::transport::Publication::GetCallbackCount ( ) const

Get the number of callbacks.

**Returns**

The number of callbacks

#### 10.142.3.6 bool gazebo::transport::Publication::GetLocallyAdvertised ( ) const

Was the topic has been advertised from this process?

**Returns**

true if the topic has been advertised from this process, false otherwise

#### 10.142.3.7 std::string gazebo::transport::Publication::GetMsgType ( ) const

Get the type of message.

**Returns**

The type of message

#### 10.142.3.8 unsigned int gazebo::transport::Publication::GetNodeCount ( ) const

Get the number of nodes.

**Returns**

The number of nodes

#### 10.142.3.9 unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ( )

Get the number of remote subscriptions.

**Returns**

The number of remote subscriptions

#### 10.142.3.10 unsigned int gazebo::transport::Publication::GetTransportCount ( ) const

Get the number of transports.

**Returns**

The number of transports

#### 10.142.3.11 bool gazebo::transport::Publication::HasTransport ( const std::string & *_host*, unsigned int *_port* )

Does a given transport exist?

**Parameters**

| | | |
|---|---|---|
| in | *_host* | Hostname of the transport |
| in | *_port* | Port of the transport |

**Returns**

true if the transport exists, false otherwise

#### 10.142.3.12 void gazebo::transport::Publication::LocalPublish ( const std::string & *_data* )

Publish data to local subscribers (skip serialization)

**Parameters**

| | | |
|---|---|---|
| in | *_data* | The data to be published |

#### 10.142.3.13 void gazebo::transport::Publication::Publish ( MessagePtr *_msg,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id* )

Publish data to remote subscribers.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to be published |
| in | *_cb* | Callback to be invoked after publishing is completed |

#### 10.142.3.14 void gazebo::transport::Publication::RemoveSubscription ( const NodePtr & *_node* )

Unsubscribe a node from our topic.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | The node |

#### 10.142.3.15 void gazebo::transport::Publication::RemoveSubscription ( const std::string & *_host,* unsigned int *_port* )

Unsubscribe a a node by host/port from our topic.

**Parameters**

| | | |
|---|---|---|
| in | *_host* | The node's hostname |
| in | *_port* | The node's port |

#### 10.142.3.16 void gazebo::transport::Publication::RemoveTransport ( const std::string & *_host,* unsigned int *_port* )

Remove a transport.

**Parameters**

| | | |
|---|---|---|
| in | *_host* | The transport's hostname |
| in | *_port* | The transport's port |

#### 10.142.3.17 void gazebo::transport::Publication::SetLocallyAdvertised ( bool *_value* )

Set whether this topic has been advertised from this process.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | If true, the topic was locally advertise, otherwise it was not |

The documentation for this class was generated from the following file:

- **Publication.hh**

## 10.143 gazebo::transport::PublicationTransport Class Reference

transport/transport.hh

`#include <PublicationTransport.hh>`

### Public Member Functions

- **PublicationTransport** (const std::string &_topic, const std::string &_msgType)
  *Constructor.*
- virtual ~**PublicationTransport** ()
  *Destructor.*
- void **AddCallback** (const boost::function< void(const std::string &)> &_cb)
  *Add a callback to the transport.*
- void **Fini** ()
  *Finalize the transport.*
- const **ConnectionPtr GetConnection** () const
  *Get the underlying connection.*
- std::string **GetMsgType** () const
  *Get the topic type.*
- std::string **GetTopic** () const
  *Get the topic name.*
- void **Init** (const **ConnectionPtr** &_conn, bool _latched)
  *Initialize the transport.*

### 10.143.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

### 10.143.2 Constructor & Destructor Documentation

#### 10.143.2.1 gazebo::transport::PublicationTransport::PublicationTransport ( const std::string & *_topic,* const std::string & *_msgType* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Topic that we're publishing |
| in | *_topic* | Type of the topic that we're publishing |

#### 10.143.2.2 virtual gazebo::transport::PublicationTransport::~PublicationTransport ( ) [virtual]

Destructor.

### 10.143.3 Member Function Documentation

#### 10.143.3.1 void gazebo::transport::PublicationTransport::AddCallback ( const boost::function< void(const std::string &)> & *_cb* )

Add a callback to the transport.

**Parameters**

| | | |
|---|---|---|
| in | *_cb* | The callback to be added |

#### 10.143.3.2 void gazebo::transport::PublicationTransport::Fini ( )

Finalize the transport.

#### 10.143.3.3 const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection ( ) const

Get the underlying connection.

**Returns**

Pointer to the underlying connection

#### 10.143.3.4 std::string gazebo::transport::PublicationTransport::GetMsgType ( ) const

Get the topic type.

**Returns**

The topic type

#### 10.143.3.5 std::string gazebo::transport::PublicationTransport::GetTopic ( ) const

Get the topic name.

**Returns**

The topic name

#### 10.143.3.6 void gazebo::transport::PublicationTransport::Init ( const ConnectionPtr & _conn, bool _latched )

Initialize the transport.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | The underlying connection. |
| in | *_latched* | True to grab the last message sent on the topic. |

The documentation for this class was generated from the following file:

- **PublicationTransport.hh**

## 10.144 gazebo::transport::Publisher Class Reference

A publisher of messages on a topic.

`#include <transport/transport.hh>`

### Public Member Functions

- **Publisher** (const std::string &_topic, const std::string &_msgType, unsigned int _limit, double _hzRate)

  *Constructor.*
- virtual ~**Publisher** ()

  *Destructor.*
- std::string **GetMsgType** () const

  *Get the message type.*
- unsigned int **GetOutgoingCount** () const

  *Get the number of outgoing messages.*
- std::string **GetPrevMsg** () const

  *Get the previously published message.*
- **MessagePtr GetPrevMsgPtr** () const

  *Get the previously published message.*
- std::string **GetTopic** () const

  *Get the topic name.*
- bool **HasConnections** () const

  *Are there any connections?*
- void **Publish** (const google::protobuf::Message &_message, bool _block=false)

  *Publish a protobuf message on the topic.*
- template<typename M >
  void **Publish** (M _message, bool _block=false)

  *Publish an arbitrary message on the topic.*
- void **SendMessage** ()

  *Send latest message over the wire. For internal use only.*
- void **SetNode** (**NodePtr** _node)

  *Set our containing node.*
- void **SetPublication** (**PublicationPtr** _publication)

  *Set the publication object for a particular publication.*

- void **WaitForConnection** () const

  *Block until a connection has been established with this publisher.*
- bool **WaitForConnection** (const **common::Time** &_timeout) const

  *Block until a connection has been established with this publisher.*

### 10.144.1 Detailed Description

A publisher of messages on a topic.

### 10.144.2 Constructor & Destructor Documentation

#### 10.144.2.1 gazebo::transport::Publisher::Publisher ( const std::string & *_topic*, const std::string & *_msgType*, unsigned int *_limit*, double *_hzRate* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Name of topic to be published |
| in | *_msgType* | Type of the message to be published |
| in | *_limit* | Maximum number of outgoing messages to queue |
| in | *_hz* | Update rate for the publisher. Units are 1.0/seconds. |

#### 10.144.2.2 virtual gazebo::transport::Publisher::~Publisher ( ) [virtual]

Destructor.

### 10.144.3 Member Function Documentation

#### 10.144.3.1 std::string gazebo::transport::Publisher::GetMsgType ( ) const

Get the message type.

**Returns**

The message type

#### 10.144.3.2 unsigned int gazebo::transport::Publisher::GetOutgoingCount ( ) const

Get the number of outgoing messages.

**Returns**

The number of outgoing messages

#### 10.144.3.3 std::string gazebo::transport::Publisher::GetPrevMsg ( ) const

Get the previously published message.

**Returns**

The previously published message, if any

#### 10.144.3.4 MessagePtr gazebo::transport::Publisher::GetPrevMsgPtr ( ) const

Get the previously published message.

**Returns**

The previously published message, if any

#### 10.144.3.5 std::string gazebo::transport::Publisher::GetTopic ( ) const

Get the topic name.

**Returns**

The topic name

#### 10.144.3.6 bool gazebo::transport::Publisher::HasConnections ( ) const

Are there any connections?

**Returns**

true if there are any connections, false otherwise

#### 10.144.3.7 void gazebo::transport::Publisher::Publish ( const google::protobuf::Message & *_message*, bool *_block* = `false` ) [inline]

Publish a protobuf message on the topic.

**Parameters**

| | | |
|---|---|---|
| in | *_message* | Message to be published |
| in | *_block* | Whether to block until the message is actually written out |

#### 10.144.3.8 template<typename M > void gazebo::transport::Publisher::Publish ( M *_message*, bool *_block* = `false` ) [inline]

Publish an arbitrary message on the topic.

**Parameters**

| | | |
|---|---|---|
| in | *_message* | Message to be published |
| in | *_block* | Whether to block until the message is actually written out |

#### 10.144.3.9 void gazebo::transport::Publisher::SendMessage ( )

Send latest message over the wire. For internal use only.

#### 10.144.3.10 void gazebo::transport::Publisher::SetNode ( NodePtr _node )

Set our containing node.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | Pointer to a node. Should be the node that create this publisher. |

#### 10.144.3.11 void gazebo::transport::Publisher::SetPublication ( PublicationPtr _publication )

Set the publication object for a particular publication.

**Parameters**

| | | |
|---|---|---|
| in | *_publication* | Pointer to the publication object to be set |

#### 10.144.3.12 void gazebo::transport::Publisher::WaitForConnection ( ) const

Block until a connection has been established with this publisher.

#### 10.144.3.13 bool gazebo::transport::Publisher::WaitForConnection ( const common::Time & _timeout ) const

Block until a connection has been established with this publisher.

**Parameters**

| | | |
|---|---|---|
| in | *_timeout* | Maxiumum time to wait. Use a negative time value to wait forever. |

**Returns**

True if a connection was established.

The documentation for this class was generated from the following file:

- **Publisher.hh**

## 10.145 QuadNode Class Reference

`#include <physics/physics.hh>`

The documentation for this class was generated from the following file:

- **MapShape.hh**

## 10.146 gazebo::math::Quaternion Class Reference

A quaternion class.

`#include <math/gzmath.hh>`

### Public Member Functions

- **Quaternion** ()

  *Default Constructor.*
- **Quaternion** (const double &_w, const double &_x, const double &_y, const double &_z)

  *Constructor.*
- **Quaternion** (const double &_roll, const double &_pitch, const double &_yaw)

  *Constructor from Euler angles in radians.*
- **Quaternion** (const **Vector3** &_axis, const double &_angle)

  *Constructor from axis angle.*
- **Quaternion** (const **Vector3** &_rpy)

  *Constructor.*
- **Quaternion** (const **Quaternion** &_qt)

  *Copy constructor.*
- ~**Quaternion** ()

  *Destructor.*
- void **Correct** ()

  *Correct any nan.*
- double **Dot** (const **Quaternion** &_q) const

  *Dot product.*
- void **GetAsAxis** (**Vector3** &_axis, double &_angle) const

  *Return rotation as axis and angle.*
- **Vector3 GetAsEuler** () const

  *Return the rotation in Euler angles.*
- **Matrix3 GetAsMatrix3** () const

  *Get the quaternion as a 3x3 matrix.*
- **Matrix4 GetAsMatrix4** () const

  *Get the quaternion as a 4x4 matrix.*
- **Quaternion GetExp** () const

  *Return the exponent.*
- **Quaternion GetInverse** () const

  *Get the inverse of this quaternion.*
- **Quaternion GetLog** () const

  *Return the logarithm.*
- double **GetPitch** ()

  *Get the Euler pitch angle in radians.*
- double **GetRoll** ()

  *Get the Euler roll angle in radians.*
- **Vector3 GetXAxis** () const

  *Return the X axis.*
- double **GetYaw** ()

*Get the Euler yaw angle in radians.*

- **Vector3 GetYAxis** () const

  *Return the Y axis.*

- **Vector3 GetZAxis** () const

  *Return the Z axis.*

- void **Invert** ()

  *Invert the quaternion.*

- bool **IsFinite** () const

  *See if a quatern is finite (e.g., not nan)*

- void **Normalize** ()

  *Normalize the quaternion.*

- bool **operator!=** (const **Quaternion** &_qt) const

  *Not equal to operator.*

- **Quaternion operator**∗ (const **Quaternion** &_q) const

  *Multiplication operator.*

- **Quaternion operator**∗ (const double &_f) const

  *Multiplication operator.*

- **Vector3 operator**∗ (const **Vector3** &_v) const

  ***Vector3*** (p. *1091) multiplication operator.*

- **Quaternion operator**∗**=** (const **Quaternion** &qt)

  *Multiplication operator.*

- **Quaternion operator+** (const **Quaternion** &_qt) const

  *Addition operator.*

- **Quaternion operator+=** (const **Quaternion** &_qt)

  *Addition operator.*

- **Quaternion operator-** (const **Quaternion** &_qt) const

  *Substraction operator.*

- **Quaternion operator-** () const

  *Unary minus operator.*

- **Quaternion operator-=** (const **Quaternion** &_qt)

  *Substraction operator.*

- **Quaternion** & **operator=** (const **Quaternion** &_qt)

  *Equal operator.*

- bool **operator==** (const **Quaternion** &_qt) const

  *Equal to operator.*

- **Vector3 RotateVector** (const **Vector3** &_vec) const

  *Rotate a vector using the quaternion.*

- **Vector3 RotateVectorReverse** (**Vector3** _vec) const

  *Do the reverse rotation of a vector by this quaternion.*

- void **Round** (int _precision)

  *Round all values to _precision decimal places.*

- void **Scale** (double _scale)

  *Scale a Quaternionion.*

- void **Set** (double _u, double _x, double _y, double _z)

  *Set this quaternion from 4 floating numbers.*

- void **SetFromAxis** (double _x, double _y, double _z, double _a)

  *Set the quaternion from an axis and angle.*

- void **SetFromAxis** (const **Vector3** &_axis, double _a)

  *Set the quaternion from an axis and angle.*
- void **SetFromEuler** (const **Vector3** &_vec)

  *Set the quaternion from Euler angles.*
- void **SetFromEuler** (double _roll, double _pitch, double _yaw)

  *Set the quaternion from Euler angles.*
- void **SetToIdentity** ()

  *Set the quatern to the identity.*

### Static Public Member Functions

- static **Quaternion EulerToQuaternion** (const **Vector3** &_vec)

  *Convert euler angles to quatern.*
- static **Quaternion EulerToQuaternion** (double _x, double _y, double _z)

  *Convert euler angles to quatern.*
- static **Quaternion Slerp** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkQ, bool _shortestPath=false)

  *Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.*
- static **Quaternion Squad** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkA, const **Quaternion** &_rkB, const **Quaternion** &_rkQ, bool _shortestPath=false)

  *Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.*

### Public Attributes

- double **w**

  *Attributes of the quaternion.*
- double **x**

  *Attributes of the quaternion.*
- double **y**

  *Attributes of the quaternion.*
- double **z**

  *Attributes of the quaternion.*

### Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Quaternion** &_q)

  *Stream insertion operator.*
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Quaternion** &_q)

  *Stream extraction operator.*

## 10.146.1 Detailed Description

A quaternion class.

## 10.146.2 Constructor & Destructor Documentation

### 10.146.2.1 gazebo::math::Quaternion::Quaternion ( )

Default Constructor.

Referenced by operator∗().

### 10.146.2.2 gazebo::math::Quaternion::Quaternion ( const double & *_w*, const double & *_x*, const double & *_y*, const double & *_z* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_w* | W param |
| in | *_x* | X param |
| in | *_y* | Y param |
| in | *_z* | Z param |

### 10.146.2.3 gazebo::math::Quaternion::Quaternion ( const double & *_roll*, const double & *_pitch*, const double & *_yaw* )

Constructor from Euler angles in radians.

**Parameters**

| | | |
|---|---|---|
| in | *_roll* | roll |
| in | *_pitch* | pitch |
| in | *_yaw* | yaw |

### 10.146.2.4 gazebo::math::Quaternion::Quaternion ( const Vector3 & *_axis*, const double & *_angle* )

Constructor from axis angle.

**Parameters**

| | | |
|---|---|---|
| in | *_axis* | the rotation axis |
| in | *_angle* | the rotation angle in radians |

### 10.146.2.5 gazebo::math::Quaternion::Quaternion ( const Vector3 & *_rpy* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_rpy* | euler angles |

### 10.146.2.6 gazebo::math::Quaternion::Quaternion ( const Quaternion & *_qt* )

Copy constructor.

**Parameters**

| | |
|---|---|
| *qt* | **Quaternion** (p. 761) to copy |

#### 10.146.2.7 gazebo::math::Quaternion::~Quaternion ( )

Destructor.

### 10.146.3 Member Function Documentation

#### 10.146.3.1 void gazebo::math::Quaternion::Correct ( ) [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

#### 10.146.3.2 double gazebo::math::Quaternion::Dot ( const Quaternion & _q ) const

Dot product.

**Parameters**

| | | |
|---|---|---|
| in | *_q* | the other quaternion |

**Returns**

the product

#### 10.146.3.3 static Quaternion gazebo::math::Quaternion::EulerToQuaternion ( const Vector3 & _vec ) [static]

Convert euler angles to quatern.

**Parameters**

| | | |
|---|---|---|
| in | | |

#### 10.146.3.4 static Quaternion gazebo::math::Quaternion::EulerToQuaternion ( double _x, double _y, double _z ) [static]

Convert euler angles to quatern.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | rotation along x |
| in | *_y* | rotation along y |
| in | *_z* | rotation along z |

#### 10.146.3.5 void gazebo::math::Quaternion::GetAsAxis ( Vector3 & *_axis,* double & *_angle* ) const

Return rotation as axis and angle.

**Parameters**

| | | |
|---|---|---|
| in | *_axis* | rotation axis |
| in | *_angle* | ccw angle in radians |

#### 10.146.3.6 Vector3 gazebo::math::Quaternion::GetAsEuler ( ) const

Return the rotation in Euler angles.

**Returns**

This quaternion as an Euler vector

#### 10.146.3.7 Matrix3 gazebo::math::Quaternion::GetAsMatrix3 ( ) const

Get the quaternion as a 3x3 matrix.

#### 10.146.3.8 Matrix4 gazebo::math::Quaternion::GetAsMatrix4 ( ) const

Get the quaternion as a 4x4 matrix.

**Returns**

a 4x4 matrix

#### 10.146.3.9 Quaternion gazebo::math::Quaternion::GetExp ( ) const

Return the exponent.

**Returns**

the exp

#### 10.146.3.10 Quaternion gazebo::math::Quaternion::GetInverse ( ) const [inline]

Get the inverse of this quaternion.

**Returns**

Inverse quarenion

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::CoordPositionSub(), gazebo::math::Pose::CoordRotationSub(), and RotateVector().

#### 10.146.3.11 Quaternion gazebo::math::Quaternion::GetLog ( ) const

Return the logarithm.

**Returns**

the log

#### 10.146.3.12 double gazebo::math::Quaternion::GetPitch ( )

Get the Euler pitch angle in radians.

**Returns**

the pitch

#### 10.146.3.13 double gazebo::math::Quaternion::GetRoll ( )

Get the Euler roll angle in radians.

**Returns**

the roll

#### 10.146.3.14 Vector3 gazebo::math::Quaternion::GetXAxis ( ) const

Return the X axis.

**Returns**

the vector

#### 10.146.3.15 double gazebo::math::Quaternion::GetYaw ( )

Get the Euler yaw angle in radians.

**Returns**

the yaw

#### 10.146.3.16 Vector3 gazebo::math::Quaternion::GetYAxis ( ) const

Return the Y axis.

**Returns**

the vector

#### 10.146.3.17 Vector3 gazebo::math::Quaternion::GetZAxis ( ) const

Return the Z axis.

**Returns**

the vector

#### 10.146.3.18 void gazebo::math::Quaternion::Invert ( )

Invert the quaternion.

#### 10.146.3.19 bool gazebo::math::Quaternion::IsFinite ( ) const

See if a quatern is finite (e.g., not nan)

**Returns**

True if quatern is finite

#### 10.146.3.20 void gazebo::math::Quaternion::Normalize ( )

Normalize the quaternion.

Referenced by gazebo::math::Pose::CoordRotationSub().

#### 10.146.3.21 bool gazebo::math::Quaternion::operator!= ( const Quaternion & _qt ) const

Not equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | **Quaternion** (p. 761) for comparison |

**Returns**

True if not equal

#### 10.146.3.22 Quaternion gazebo::math::Quaternion::operator* ( const Quaternion & _q ) const [inline]

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | **Quaternion** (p. 761) for multiplication |

**Returns**

This quaternion multiplied by the parameter

References Quaternion(), w, x, y, and z.

#### 10.146.3.23 Quaternion gazebo::math::Quaternion::operator∗ ( const double & _f ) const

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_f* | factor |

**Returns**

quaternion multiplied by _f

#### 10.146.3.24 Vector3 gazebo::math::Quaternion::operator∗ ( const Vector3 & _v ) const

**Vector3** (p. 1091) multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | vector to multiply |

#### 10.146.3.25 Quaternion gazebo::math::Quaternion::operator∗= ( const Quaternion & *qt* )

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | **Quaternion** (p. 761) for multiplication |

**Returns**

This quatern multiplied by the parameter

#### 10.146.3.26 Quaternion gazebo::math::Quaternion::operator+ ( const Quaternion & _qt ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | quaternion for addition |

**Returns**

this quaternion + _qt

#### 10.146.3.27 Quaternion gazebo::math::Quaternion::operator+= ( const Quaternion & _qt )

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | quaternion for addition |

**Returns**

this quaternion + qt

#### 10.146.3.28 Quaternion gazebo::math::Quaternion::operator- ( const Quaternion & _qt ) const

Substraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | quaternion to substract |

**Returns**

this quaternion - _qt

#### 10.146.3.29 Quaternion gazebo::math::Quaternion::operator- ( ) const

Unary minus operator.

**Returns**

negates each component of the quaternion

#### 10.146.3.30 Quaternion gazebo::math::Quaternion::operator-= ( const Quaternion & _qt )

Substraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | **Quaternion** (p. 761) for substraction |

**Returns**

This quatern - qt

#### 10.146.3.31 Quaternion& gazebo::math::Quaternion::operator= ( const Quaternion & _qt )

Equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | **Quaternion** (p. 761) to copy |

#### 10.146.3.32 bool gazebo::math::Quaternion::operator== ( const Quaternion & *_qt* ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_qt* | **Quaternion** (p. 761) for comparison |

**Returns**

True if equal

#### 10.146.3.33 Vector3 gazebo::math::Quaternion::RotateVector ( const Vector3 & *_vec* ) const [inline]

Rotate a vector using the quaternion.

**Parameters**

| | | |
|---|---|---|
| in | *_vec* | vector to rotate |

**Returns**

the rotated vector

References GetInverse(), gazebo::math::Vector3::x, x, gazebo::math::Vector3::y, y, gazebo::math::Vector3::z, and z.

#### 10.146.3.34 Vector3 gazebo::math::Quaternion::RotateVectorReverse ( Vector3 *_vec* ) const

Do the reverse rotation of a vector by this quaternion.

**Parameters**

| | | |
|---|---|---|
| in | *_vec* | the vector |

**Returns**

the

#### 10.146.3.35 void gazebo::math::Quaternion::Round ( int *_precision* )

Round all values to _precision decimal places.

**Parameters**

| | | |
|---|---|---|
| in | *_precision* | the precision |

#### 10.146.3.36 void gazebo::math::Quaternion::Scale ( double *_scale* )

Scale a Quaternionion.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Amount to scale this rotation |

#### 10.146.3.37 void gazebo::math::Quaternion::Set ( double *_u*, double *_x*, double *_y*, double *_z* )

Set this quaternion from 4 floating numbers.

**Parameters**

| | | |
|---|---|---|
| in | *_u* | u |
| in | *_x* | x |
| in | *_y* | y |
| in | *_z* | z |

#### 10.146.3.38 void gazebo::math::Quaternion::SetFromAxis ( double *_x*, double *_y*, double *_z*, double *_a* )

Set the quaternion from an axis and angle.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | X axis |
| in | *_y* | Y axis |
| in | *_z* | Z axis |
| in | *_a* | **Angle** (p. 137) in radians |

#### 10.146.3.39 void gazebo::math::Quaternion::SetFromAxis ( const Vector3 & *_axis*, double *_a* )

Set the quaternion from an axis and angle.

**Parameters**

| | | |
|---|---|---|
| in | *_axis* | Axis |
| in | *_a* | **Angle** (p. 137) in radians |

#### 10.146.3.40 void gazebo::math::Quaternion::SetFromEuler ( const Vector3 & *_vec* )

Set the quaternion from Euler angles.

The order of operations are roll, pitch, yaw.

**Parameters**

| | | |
|---|---|---|
| in | *vec* | Euler angle |

#### 10.146.3.41 void gazebo::math::Quaternion::SetFromEuler ( double _roll, double _pitch, double _yaw )

Set the quaternion from Euler angles.

**Parameters**

| | | |
|---|---|---|
| in | *_roll* | Roll angle (radians). |
| in | *_pitch* | Roll angle (radians). |
| in | *_yaw* | Roll angle (radians). |

#### 10.146.3.42 void gazebo::math::Quaternion::SetToIdentity ( )

Set the quatern to the identity.

#### 10.146.3.43 static Quaternion gazebo::math::Quaternion::Slerp ( double _fT, const Quaternion & _rkP, const Quaternion & _rkQ, bool _shortestPath = false ) [static]

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

**Parameters**

| | | |
|---|---|---|
| in | *_ft* | the interpolation parameter |
| in | *_rkP* | the beginning quaternion |
| in | *_rkQ* | the end quaternion |
| in | *_shortestPath* | when true, the rotation may be inverted to get to minimize rotation |

#### 10.146.3.44 static Quaternion gazebo::math::Quaternion::Squad ( double _fT, const Quaternion & _rkP, const Quaternion & _rkA, const Quaternion & _rkB, const Quaternion & _rkQ, bool _shortestPath = false ) [static]

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

**Parameters**

| | | |
|---|---|---|
| in | *_ft* | the interpolation parameter |
| in | *_rkP* | the beginning quaternion |
| in | *_rkA* | first intermediate quaternion |
| in | *_rkB* | second intermediate quaternion |
| in | *_rkQ* | the end quaternion |
| in | *_shortestPath* | when true, the rotation may be inverted to get to minimize rotation |

### 10.146.4 Friends And Related Function Documentation

#### 10.146.4.1 std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Quaternion & _q ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_q* | quaternion to output |

**Returns**

the stream

#### 10.146.4.2 std::istream& operator>> ( std::istream & *_in,* gazebo::math::Quaternion & *_q* ) [friend]

Stream extraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_in* | input stream |
| in | *_q* | **Quaternion** (p. 761) to read values into |

**Returns**

The istream

### 10.146.5 Member Data Documentation

#### 10.146.5.1 double gazebo::math::Quaternion::w

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), Correct(), GetInverse(), and operator∗().

#### 10.146.5.2 double gazebo::math::Quaternion::x

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

#### 10.146.5.3 double gazebo::math::Quaternion::y

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

#### 10.146.5.4 double gazebo::math::Quaternion::z

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

The documentation for this class was generated from the following file:

- **Quaternion.hh**

## 10.147 gazebo::math::Rand Class Reference

Random number generator class.

`#include <gzmath/gzmath.hh>`

### Static Public Member Functions

- static double **GetDblNormal** (double _mean=0, double _sigma=1)

  *Get a double from a normal distribution.*
- static double **GetDblUniform** (double _min=0, double _max=1)

  *Get a double from a uniform distribution.*
- static int **GetIntNormal** (int _mean, int _sigma)

  *Get a double from a normal distribution.*
- static int **GetIntUniform** (int _min, int _max)

  *Get a integer from a uniform distribution.*
- static uint32_t **GetSeed** ()

  *Get the seed value.*
- static void **SetSeed** (uint32_t _seed)

  *Set the seed value.*

### 10.147.1 Detailed Description

Random number generator class.

### 10.147.2 Member Function Documentation

#### 10.147.2.1 static double gazebo::math::Rand::GetDblNormal ( double *_mean* = 0, double *_sigma* = 1 ) [static]

Get a double from a normal distribution.

**Parameters**

| | | |
|---|---|---|
| in | *_mean* | Mean value for the distribution |
| in | *_sigma* | Sigma value for the distribution |

#### 10.147.2.2 static double gazebo::math::Rand::GetDblUniform ( double *_min* = 0, double *_max* = 1 ) [static]

Get a double from a uniform distribution.

**Parameters**

| | | |
|---|---|---|
| in | *_min* | Minimum bound for the random number |
| in | *_max* | Maximum bound for the random number |

#### 10.147.2.3 static int gazebo::math::Rand::GetIntNormal ( int *_mean*, int *_sigma* ) [static]

Get a double from a normal distribution.

**Parameters**

| | | |
|---|---|---|
| in | *_mean* | Mean value for the distribution |
| in | *_sigma* | Sigma value for the distribution |

#### 10.147.2.4 static int gazebo::math::Rand::GetIntUniform ( int *_min*, int *_max* ) [static]

Get a integer from a uniform distribution.

**Parameters**

| | | |
|---|---|---|
| in | *_min* | Minimum bound for the random number |
| in | *_max* | Maximum bound for the random number |

#### 10.147.2.5 static uint32_t gazebo::math::Rand::GetSeed ( ) [static]

Get the seed value.

**Returns**

The seed value used to initialize the random number generator.

#### 10.147.2.6 static void gazebo::math::Rand::SetSeed ( uint32_t *_seed* ) [static]

Set the seed value.

**Parameters**

| | | |
|---|---|---|
| in | *_seed* | The seed used to initialize the randon number generator. |

The documentation for this class was generated from the following file:

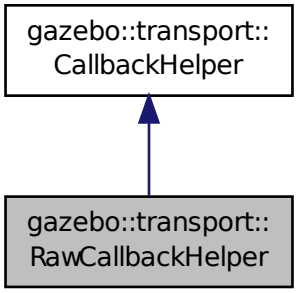
- **Rand.hh**

## 10.148 gazebo::transport::RawCallbackHelper Class Reference

Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

`#include <CallbackHelper.hh>`

Inheritance diagram for gazebo::transport::RawCallbackHelper:




## Public Member Functions

- **RawCallbackHelper** (const boost::function< void(const std::string &)> &_cb, bool _latching=false)

  *Constructor.*
- std::string **GetMsgType** () const

  *Get the typename of the message that is handled.*
- virtual bool **HandleData** (const std::string &_newdata, boost::function< void(uint32_t)> _cb, uint32_t _id)

  *Process new incoming data.*
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  *Process new incoming message.*
- virtual bool **IsLocal** () const

  *Is the callback local?*

## Additional Inherited Members

### 10.148.1 Detailed Description

Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

Raw means that the data has not been converted into a protobuf message.

### 10.148.2 Constructor & Destructor Documentation

#### 10.148.2.1 gazebo::transport::RawCallbackHelper::RawCallbackHelper ( const boost::function< void(const std::string &)> & *_cb*, bool *_latching* = `false` ) [inline]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_cb* | boost function to call on incoming messages |
| in | *_latching* | Set to true to make the callback helper latching. |

### 10.148.3 Member Function Documentation

#### 10.148.3.1 std::string gazebo::transport::RawCallbackHelper::GetMsgType ( ) const [inline], [virtual]

Get the typename of the message that is handled.

**Returns**

String representation of the message type

Reimplemented from **gazebo::transport::CallbackHelper** (p. 182).

#### 10.148.3.2 virtual bool gazebo::transport::RawCallbackHelper::HandleData ( const std::string & *_newdata,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id* ) [inline], [virtual]

Process new incoming data.

**Parameters**

| | | |
|---|---|---|
| in | *_newdata* | Incoming data to be processed |

**Returns**

true if successfully processed; false otherwise

**Parameters**

| | | |
|---|---|---|
| in | *_cb* | If non-null, callback to be invoked which signals that transmission is complete. |
| in | *_id* | ID associated with the message data. |

Implements **gazebo::transport::CallbackHelper** (p. 182).

#### 10.148.3.3 virtual bool gazebo::transport::RawCallbackHelper::HandleMessage ( MessagePtr *_newMsg* ) [inline], [virtual]

Process new incoming message.

**Parameters**

| | | |
|---|---|---|
| in | *_newMsg* | Incoming message to be processed |

**Returns**

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 183).

#### 10.148.3.4 virtual bool gazebo::transport::RawCallbackHelper::IsLocal ( ) const [inline], [virtual]

Is the callback local?

**Returns**

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 183).

The documentation for this class was generated from the following file:

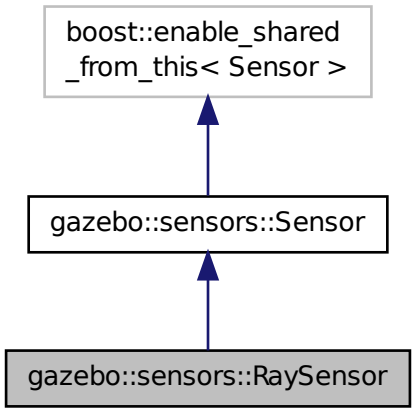
- **CallbackHelper.hh**

## 10.149 gazebo::sensors::RaySensor Class Reference

**Sensor** (p. 837) with one or more rays.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RaySensor:




### Public Member Functions

- **RaySensor** ()

  *Constructor.*
- virtual ~**RaySensor** ()

  *Destructor.*
- **math::Angle GetAngleMax** () const

  *Get the maximum angle.*
- **math::Angle GetAngleMin** () const

  *Get the minimum angle.*
- double **GetAngleResolution** () const

  *Get the angle in radians between each range.*
- int **GetFiducial** (int _index)

*Get detected fiducial value for a ray.*

- **physics::MultiRayShapePtr GetLaserShape** () const

  *Returns a pointer to the internal **physics::MultiRayShape** (p. 664).*

- double **GetRange** (int _index)

  *Get detected range for a ray.*

- int **GetRangeCount** () const

  *Get the range count.*

- double **GetRangeMax** () const

  *Get the maximum range.*

- double **GetRangeMin** () const

  *Get the minimum range.*

- double **GetRangeResolution** () const

  *Get the range resolution.*

- void **GetRanges** (std::vector< double > &_ranges)

  *Get all the ranges.*

- int **GetRayCount** () const

  *Get the ray count.*

- double **GetRetro** (int _index)

  *Get detected retro (intensity) value for a ray.*

- virtual std::string **GetTopic** () const

  *Returns the topic name as set in SDF.*

- **math::Angle GetVerticalAngleMax** () const

  *Get the vertical scan line top angle.*

- **math::Angle GetVerticalAngleMin** () const

  *Get the vertical scan bottom angle.*

- int **GetVerticalRangeCount** () const

  *Get the vertical scan line count.*

- int **GetVerticalRayCount** () const

  *Get the vertical scan line count.*

- virtual void **Init** ()

  *Initialize the sensor.*

- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*

- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

## Protected Member Functions

- virtual void **Fini** ()

  *Finalize the sensor.*

- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Additional Inherited Members

## 10.149.1 Detailed Description

**Sensor** (p. 837) with one or more rays.

This sensor cast rays into the world, tests for intersections, and reports the range to the nearest object. It is used by ranging sensor models (e.g., sonars and scanning laser range finders).

## 10.149.2 Constructor & Destructor Documentation

#### 10.149.2.1 gazebo::sensors::RaySensor::RaySensor ( )

Constructor.

#### 10.149.2.2 virtual gazebo::sensors::RaySensor::~RaySensor ( ) [virtual]

Destructor.

## 10.149.3 Member Function Documentation

#### 10.149.3.1 virtual void gazebo::sensors::RaySensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.149.3.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax ( ) const

Get the maximum angle.

**Returns**

the maximum angle object

#### 10.149.3.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin ( ) const

Get the minimum angle.

**Returns**

The minimum angle object

#### 10.149.3.4 double gazebo::sensors::RaySensor::GetAngleResolution ( ) const

Get the angle in radians between each range.

**Returns**

Resolution of the angle

#### 10.149.3.5 int gazebo::sensors::RaySensor::GetFiducial ( int _index )

Get detected fiducial value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index value of specific ray |

**Returns**

Fiducial value

#### 10.149.3.6 physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape ( ) const [inline]

Returns a pointer to the internal **physics::MultiRayShape** (p. 664).

**Returns**

Pointer to ray shape

#### 10.149.3.7 double gazebo::sensors::RaySensor::GetRange ( int _index )

Get detected range for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of specific ray |

**Returns**

Returns DBL_MAX for no detection.

#### 10.149.3.8 int gazebo::sensors::RaySensor::GetRangeCount ( ) const

Get the range count.

**Returns**

The number of ranges

#### 10.149.3.9 double gazebo::sensors::RaySensor::GetRangeMax ( ) const

Get the maximum range.

**Returns**

The maximum range

#### 10.149.3.10 double gazebo::sensors::RaySensor::GetRangeMin ( ) const

Get the minimum range.

**Returns**

The minimum range

#### 10.149.3.11 double gazebo::sensors::RaySensor::GetRangeResolution ( ) const

Get the range resolution.

**Returns**

Resolution of the range

#### 10.149.3.12 void gazebo::sensors::RaySensor::GetRanges ( std::vector< double > & *_ranges* )

Get all the ranges.

**Parameters**

| | |
|---|---|
| *_ranges* | A vector that will contain all the range data |

#### 10.149.3.13 int gazebo::sensors::RaySensor::GetRayCount ( ) const

Get the ray count.

**Returns**

The number of rays

#### 10.149.3.14 double gazebo::sensors::RaySensor::GetRetro ( int *_index* )

Get detected retro (intensity) value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of specific ray |

**Returns**

Retro (intensity) value for ray

#### 10.149.3.15 virtual std::string gazebo::sensors::RaySensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

**Returns**

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 843).

#### 10.149.3.16 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

**Returns**

The Maximum angle of the scan block

#### 10.149.3.17 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

**Returns**

The minimum angle of the scan block

#### 10.149.3.18 int gazebo::sensors::RaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

**Returns**

The number of scan lines vertically

#### 10.149.3.19 int gazebo::sensors::RaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

**Returns**

The number of scan lines vertically

#### 10.149.3.20 virtual void gazebo::sensors::RaySensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.149.3.21 virtual bool gazebo::sensors::RaySensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.149.3.22 virtual void gazebo::sensors::RaySensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.149.3.23 virtual void gazebo::sensors::RaySensor::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

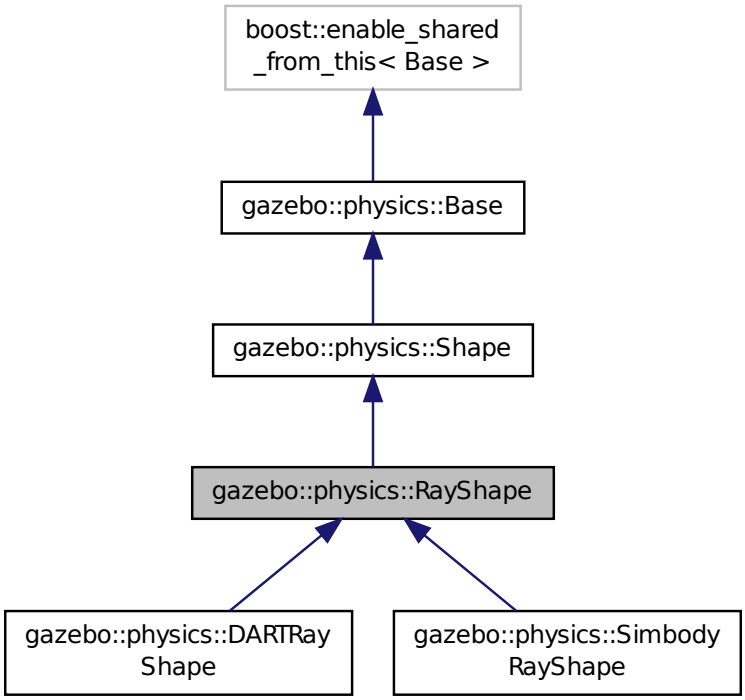
- **RaySensor.hh**

## 10.150 gazebo::physics::RayShape Class Reference

**Base** (p. 159) class for Ray collision geometry.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::RayShape:




### Public Member Functions

- **RayShape** (**PhysicsEnginePtr** _physicsEngine)

  *Constructor for a global ray.*
- **RayShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**RayShape** ()

  *Destructor.*
- void **FillMsg** (msgs::Geometry &_msg)

  *Fill a message with data from this object.*
- int **GetFiducial** () const

  *Get the fiducial id detected by this ray.*
- virtual void **GetGlobalPoints** (**math::Vector3** &_posA, **math::Vector3** &_posB)

  *Get the global starting and ending points.*

- virtual void **GetIntersection** (double &_dist, std::string &_entity)=0

  *Get the nearest intersection.*
- double **GetLength** () const

  *Get the length of the ray.*
- virtual void **GetRelativePoints** (**math::Vector3** &_posA, **math::Vector3** &_posB)

  *Get the relative starting and ending points.*
- float **GetRetro** () const

  *Get the retro-reflectivness detected by this ray.*
- virtual void **Init** ()

  *In the ray.*
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *Update this shape from a message.*
- void **SetFiducial** (int _fid)

  *Set the fiducial id detected by this ray.*
- virtual void **SetLength** (double _len)

  *Set the length of the ray.*
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)

  *Set the ray based on starting and ending points relative to the body.*
- void **SetRetro** (float _retro)

  *Set the retro-reflectivness detected by this ray.*
- virtual void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the ray.*
- virtual void **Update** ()=0

  *Update the ray collision.*

## Protected Attributes

- int **contactFiducial**

  *Fiducial ID value.*
- double **contactLen**

  *Length of the ray.*
- double **contactRetro**

  *Retro reflectance value.*
- **math::Vector3 globalEndPos**

  *End position of the ray in global cs.*
- **math::Vector3 globalStartPos**

  *Start position of the ray in global cs.*
- **math::Vector3 relativeEndPos**

  *End position of the ray, relative to the body.*
- **math::Vector3 relativeStartPos**

  *Start position of the ray, relative to the body.*

## Additional Inherited Members

### 10.150.1 Detailed Description

**Base** (p. 159) class for Ray collision geometry.

### 10.150.2 Constructor & Destructor Documentation

#### 10.150.2.1 gazebo::physics::RayShape::RayShape ( PhysicsEnginePtr _physicsEngine ) [explicit]

Constructor for a global ray.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Pointer to the physics engine. |

#### 10.150.2.2 gazebo::physics::RayShape::RayShape ( CollisionPtr _parent ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | **Collision** (p. 220) parent of the shape. |

#### 10.150.2.3 virtual gazebo::physics::RayShape::~RayShape ( ) [virtual]

Destructor.

### 10.150.3 Member Function Documentation

#### 10.150.3.1 void gazebo::physics::RayShape::FillMsg ( msgs::Geometry & _msg ) [virtual]

Fill a message with data from this object.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message to fill. Implement this function. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.150.3.2 int gazebo::physics::RayShape::GetFiducial ( ) const

Get the fiducial id detected by this ray.

**Returns**

Fiducial id detected.

#### 10.150.3.3 virtual void gazebo::physics::RayShape::GetGlobalPoints ( math::Vector3 & _posA, math::Vector3 & _posB ) [virtual]

Get the global starting and ending points.

**Parameters**

| | | |
|---|---|---|
| out | *_posA* | Returns the starting point. |
| out | *_posB* | Returns the ending point. |

#### 10.150.3.4 virtual void gazebo::physics::RayShape::GetIntersection ( double & _dist, std::string & _entity ) [pure virtual]

Get the nearest intersection.

**Parameters**

| | | |
|---|---|---|
| out | *_dist* | Distance to the intersection. |
| out | *_entity* | Name of the entity the ray intersected with. |

Implemented in **gazebo::physics::DARTRayShape** (p. 75), and **gazebo::physics::SimbodyRayShape** (p. 928).

#### 10.150.3.5 double gazebo::physics::RayShape::GetLength ( ) const

Get the length of the ray.

**Returns**

The ray length.

#### 10.150.3.6 virtual void gazebo::physics::RayShape::GetRelativePoints ( math::Vector3 & _posA, math::Vector3 & _posB ) [virtual]

Get the relative starting and ending points.

**Parameters**

| | | |
|---|---|---|
| in | *_posA* | Returns the starting point. |
| in | *_posB* | Returns the ending point. |

#### 10.150.3.7 float gazebo::physics::RayShape::GetRetro ( ) const

Get the retro-reflectivness detected by this ray.

**Returns**

Retro reflectance value.

#### 10.150.3.8 virtual void gazebo::physics::RayShape::Init ( ) [virtual]

In the ray.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.150.3.9 virtual void gazebo::physics::RayShape::ProcessMsg ( const msgs::Geometry & _msg ) [virtual]

Update this shape from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to update from. Implement this function. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.150.3.10 void gazebo::physics::RayShape::SetFiducial ( int *_fid* )

Set the fiducial id detected by this ray.

**Parameters**

| | | |
|---|---|---|
| in | *_fid* | Fiducial id detected by this ray. |

#### 10.150.3.11 virtual void gazebo::physics::RayShape::SetLength ( double *_len* ) [virtual]

Set the length of the ray.

**Parameters**

| | | |
|---|---|---|
| in | *_len* | Length of the array. |

#### 10.150.3.12 virtual void gazebo::physics::RayShape::SetPoints ( const math::Vector3 & *_posStart*, const math::Vector3 & *_posEnd* ) [virtual]

Set the ray based on starting and ending points relative to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_posStart* | Start position, relative the body. |
| in | *_posEnd* | End position, relative to the body. |

Reimplemented in **gazebo::physics::DARTRayShape** (p. 75), and **gazebo::physics::SimbodyRayShape** (p. 928).

#### 10.150.3.13 void gazebo::physics::RayShape::SetRetro ( float *_retro* )

Set the retro-reflectivness detected by this ray.

**Parameters**

| | | |
|---|---|---|
| in | *_retro* | Retro reflectance value. |

#### 10.150.3.14 virtual void gazebo::physics::RayShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set the scale of the ray.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.150.3.15 virtual void gazebo::physics::RayShape::Update ( ) [pure virtual]

Update the ray collision.

Reimplemented from **gazebo::physics::Base** (p. 171).

Implemented in **gazebo::physics::DARTRayShape** (p. 75), and **gazebo::physics::SimbodyRayShape** (p. 929).

### 10.150.4 Member Data Documentation

#### 10.150.4.1 int gazebo::physics::RayShape::contactFiducial [protected]

Fiducial ID value.

#### 10.150.4.2 double gazebo::physics::RayShape::contactLen [protected]

Length of the ray.

#### 10.150.4.3 double gazebo::physics::RayShape::contactRetro [protected]

Retro reflectance value.

#### 10.150.4.4 math::Vector3 gazebo::physics::RayShape::globalEndPos [protected]

End position of the ray in global cs.

#### 10.150.4.5 math::Vector3 gazebo::physics::RayShape::globalStartPos [protected]

Start position of the ray in global cs.

#### 10.150.4.6 math::Vector3 gazebo::physics::RayShape::relativeEndPos [protected]

End position of the ray, relative to the body.

#### 10.150.4.7 math::Vector3 gazebo::physics::RayShape::relativeStartPos [protected]

Start position of the ray, relative to the body.

The documentation for this class was generated from the following file:

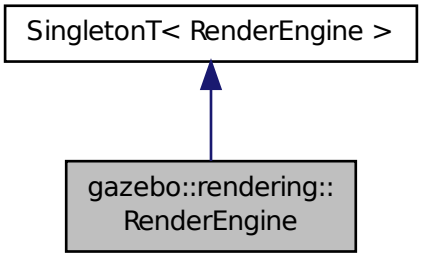
- **RayShape.hh**

## 10.151 gazebo::rendering::RenderEngine Class Reference

Adaptor to Ogre3d.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RenderEngine:




## Public Types

- enum **RenderPathType** {
  **NONE** = 0, **VERTEX** = 1, **FORWARD** = 2, **DEFERRED** = 3,
  **RENDER_PATH_COUNT** }

  *The type of rendering path used by the rendering engine.*

## Public Member Functions

- void **AddResourcePath** (const std::string &_uri)

  *Add a new path for **Ogre** (p. 130) to search for resources.*
- **ScenePtr CreateScene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)

  *Create a scene.*
- void **Fini** ()

  *Tears down the rendering engine.*
- **RenderPathType GetRenderPathType** () const

  *Get the type of rendering path to use.*
- **ScenePtr GetScene** (const std::string &_name="")

  *Get a scene by name.*
- **ScenePtr GetScene** (unsigned int _index)

  *Get a scene by index.*
- unsigned int **GetSceneCount** () const

  *Get the number of scenes.*
- **WindowManagerPtr GetWindowManager** () const

  *Get a pointer to the window manager.*
- void **Init** ()

  *Initialize **Ogre** (p. 130). Load must happen before Init.*
- void **Load** ()

  *Load the parameters for **Ogre** (p. 130). Load must happen before Init.*
- void **RemoveScene** (const std::string &_name)

  *Remove a scene.*

### Public Attributes

- Ogre::Root ∗ **root**

  *Pointer to the root scene node.*

### Protected Attributes

- void ∗ **dummyContext**

  *GLX context used to render the scenes.Used for gui-less operation.*
- void ∗ **dummyDisplay**

  *Pointer to the dummy display.Used for gui-less operation.*
- uint64_t **dummyWindowId**

  *ID for a dummy window. Used for gui-less operation.*

### Additional Inherited Members

## 10.151.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

## 10.151.2 Member Enumeration Documentation

### 10.151.2.1 enum gazebo::rendering::RenderEngine::RenderPathType

The type of rendering path used by the rendering engine.

**Enumerator**

***NONE*** No rendering is done.

***VERTEX*** Most basic rendering, with least fidelity.

***FORWARD*** Utilizes the RTT shader system.

***DEFERRED*** Utilizes deferred rendering. Best fidelity.

***RENDER_PATH_COUNT*** Count of the rendering path enums.

## 10.151.3 Member Function Documentation

### 10.151.3.1 void gazebo::rendering::RenderEngine::AddResourcePath ( const std::string & *_uri* )

Add a new path for **Ogre** (p. 130) to search for resources.

**Parameters**

| | | |
|---|---|---|
| in | *_uri* | URI of the path. The uri should be of the form `file://` or model:// |

#### 10.151.3.2 ScenePtr gazebo::rendering::RenderEngine::CreateScene ( const std::string & *_name*, bool *_enableVisualizations*, bool *_isServer* = false )

Create a scene.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the scene. |
| in | *_enable-Visualizations* | True enables visualization elements such as laser lines. |

#### 10.151.3.3 void gazebo::rendering::RenderEngine::Fini ( )

Tears down the rendering engine.

#### 10.151.3.4 RenderPathType gazebo::rendering::RenderEngine::GetRenderPathType ( ) const

Get the type of rendering path to use.

This is automatically determined based on the computers capabilities

**Returns**

The RenderPathType

#### 10.151.3.5 ScenePtr gazebo::rendering::RenderEngine::GetScene ( const std::string & *_name* = " " )

Get a scene by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the scene to retreive. |

**Returns**

A pointer to the **Scene** (p. 814), or NULL if the scene doesn't exist.

#### 10.151.3.6 ScenePtr gazebo::rendering::RenderEngine::GetScene ( unsigned int *_index* )

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 795).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | The index of the scene. |

**Returns**

A pointer to a **Scene** (p. 814), or NULL if the index was invalid.

#### 10.151.3.7 unsigned int gazebo::rendering::RenderEngine::GetSceneCount ( ) const

Get the number of scenes.

**Returns**

The number of scenes created by the **RenderEngine** (p. 791).

#### 10.151.3.8 WindowManagerPtr gazebo::rendering::RenderEngine::GetWindowManager ( ) const

Get a pointer to the window manager.

**Returns**

Pointer to the window manager.

#### 10.151.3.9 void gazebo::rendering::RenderEngine::Init ( )

Initialize **Ogre** (p. 130). Load must happen before Init.

#### 10.151.3.10 void gazebo::rendering::RenderEngine::Load ( )

Load the parameters for **Ogre** (p. 130). Load must happen before Init.

#### 10.151.3.11 void gazebo::rendering::RenderEngine::RemoveScene ( const std::string & *_name* )

Remove a scene.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the scene to remove. |

### 10.151.4 Member Data Documentation

#### 10.151.4.1 void* gazebo::rendering::RenderEngine::dummyContext [protected]

GLX context used to render the scenes.Used for gui-less operation.

#### 10.151.4.2 void* gazebo::rendering::RenderEngine::dummyDisplay [protected]

Pointer to the dummy display.Used for gui-less operation.

#### 10.151.4.3 uint64_t gazebo::rendering::RenderEngine::dummyWindowId [protected]

ID for a dummy window. Used for gui-less operation.

#### 10.151.4.4 Ogre::Root* gazebo::rendering::RenderEngine::root

Pointer to the root scene node.

The documentation for this class was generated from the following file:

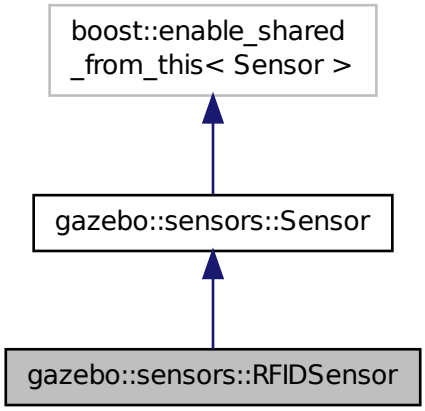
- **RenderEngine.hh**

## 10.152 gazebo::sensors::RFIDSensor Class Reference

**Sensor** (p. 837) class for RFID type of sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RFIDSensor:




### Public Member Functions

- **RFIDSensor** ()
  *Constructor.*
- virtual ~**RFIDSensor** ()
  *Destructor.*
- void **AddTag** (**RFIDTag** *_tag)
- virtual void **Fini** ()
  *Finalize the sensor.*
- virtual void **Init** ()
  *Initialize the sensor.*
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
  *Load the sensor with SDF parameters.*
- virtual void **Load** (const std::string &_worldName)
  *Load the sensor with default parameters.*

### Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Additional Inherited Members

## 10.152.1 Detailed Description

**Sensor** (p. 837) class for RFID type of sensor.

## 10.152.2 Constructor & Destructor Documentation

#### 10.152.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ( )

Constructor.

#### 10.152.2.2 virtual gazebo::sensors::RFIDSensor::~RFIDSensor ( ) [virtual]

Destructor.

## 10.152.3 Member Function Documentation

#### 10.152.3.1 void gazebo::sensors::RFIDSensor::AddTag ( RFIDTag * *_tag* )

#### 10.152.3.2 virtual void gazebo::sensors::RFIDSensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.152.3.3 virtual void gazebo::sensors::RFIDSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.152.3.4 virtual void gazebo::sensors::RFIDSensor::Load ( const std::string & *_worldName*, sdf::ElementPtr *_sdf* ) [virtual]

Load the sensor with SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF **Sensor** (p. 837) parameters. |
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.152.3.5 virtual void gazebo::sensors::RFIDSensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.152.3.6 virtual void gazebo::sensors::RFIDSensor::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

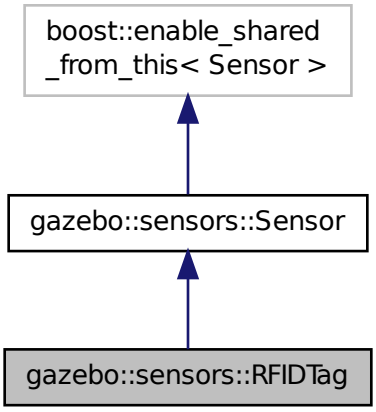
- **RFIDSensor.hh**

## 10.153 gazebo::sensors::RFIDTag Class Reference

**RFIDTag** (p. 798) to interact with RFIDTagSensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RFIDTag:




### Public Member Functions

- **RFIDTag** ()

  *Constructor.*
- virtual ~**RFIDTag** ()

  *Destructor.*
- virtual void **Fini** ()

  *Finalize the sensor.*
- **math::Pose GetTagPose** () const

  *Returns pose of tag in world coordinate.*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr &_sdf)
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

### Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Additional Inherited Members

### 10.153.1 Detailed Description

**RFIDTag** (p. 798) to interact with RFIDTagSensors.

### 10.153.2 Constructor & Destructor Documentation

#### 10.153.2.1 gazebo::sensors::RFIDTag::RFIDTag ( )

Constructor.

#### 10.153.2.2 virtual gazebo::sensors::RFIDTag::~RFIDTag ( ) [virtual]

Destructor.

### 10.153.3 Member Function Documentation

#### 10.153.3.1 virtual void gazebo::sensors::RFIDTag::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.153.3.2 math::Pose gazebo::sensors::RFIDTag::GetTagPose ( ) const [inline]

Returns pose of tag in world coordinate.

**Returns**

Pose of object.

#### 10.153.3.3 virtual void gazebo::sensors::RFIDTag::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.153.3.4 virtual void gazebo::sensors::RFIDTag::Load ( const std::string & *_worldName,* sdf::ElementPtr & *_sdf* ) [virtual]

#### 10.153.3.5 virtual void gazebo::sensors::RFIDTag::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.153.3.6 virtual void gazebo::sensors::RFIDTag::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

The documentation for this class was generated from the following file:

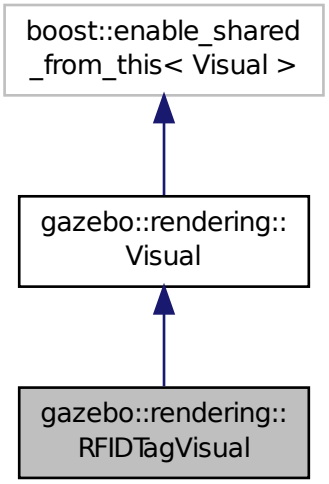
- **RFIDTag.hh**

## 10.154 gazebo::rendering::RFIDTagVisual Class Reference

Visualization for RFID tags sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RFIDTagVisual:




### Public Member Functions

- **RFIDTagVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  *Constructor.*
- virtual ~**RFIDTagVisual** ()

  *Destructor.*

### Additional Inherited Members

### 10.154.1 Detailed Description

Visualization for RFID tags sensor.

### 10.154.2 Constructor & Destructor Documentation

#### 10.154.2.1 gazebo::rendering::RFIDTagVisual::RFIDTagVisual ( const std::string & *_name,* VisualPtr *_vis,* const std::string & *_topicName* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual. |
| in | *_vis* | Parent visual. |
| in | *_topicName* | Name of the topic that publishes RFID data. |

**See Also**

**sensors::RFIDSensor** (p. 796)

#### 10.154.2.2 virtual gazebo::rendering::RFIDTagVisual::~RFIDTagVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

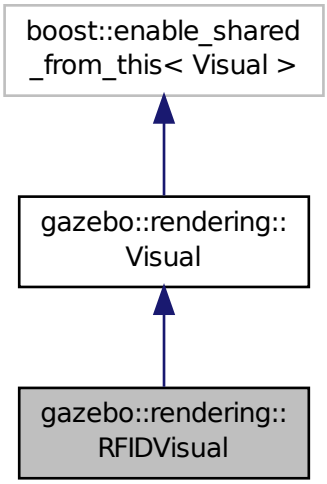
- **RFIDTagVisual.hh**

## 10.155 gazebo::rendering::RFIDVisual Class Reference

Visualization for RFID sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RFIDVisual:




## Public Member Functions

- **RFIDVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  *Constructor.*
- virtual ~**RFIDVisual** ()

  *Destructor.*

## Additional Inherited Members

### 10.155.1 Detailed Description

Visualization for RFID sensor.

### 10.155.2 Constructor & Destructor Documentation

#### 10.155.2.1 gazebo::rendering::RFIDVisual::RFIDVisual ( const std::string & *_name*, VisualPtr *_vis*, const std::string & *_topicName* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the **Visual** (p. 1121). |
| in | *_vis* | Parent **Visual** (p. 1121). |
| in | *_topicName* | Name of the topic which publishes RFID data. |

#### 10.155.2.2 virtual gazebo::rendering::RFIDVisual::~RFIDVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **RFIDVisual.hh**

## 10.156 Road Class Reference

Used to render a strip of road.

`#include <rendering/rendering.hh>`

### 10.156.1 Detailed Description

Used to render a strip of road.

The documentation for this class was generated from the following file:

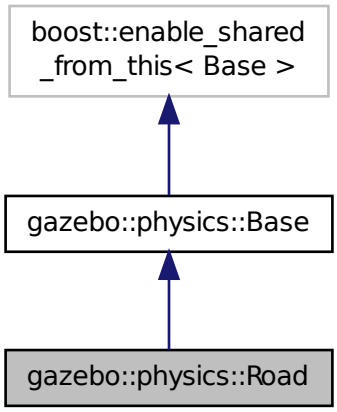
- **Road2d.hh**

## 10.157 gazebo::physics::Road Class Reference

for building a **Road** (p. 804) from SDF

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Road:



### Public Member Functions

- **Road** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**Road** ()

  *Destructor.*
- virtual void **Init** ()

  *Initialize the road.*
- void **Load** (sdf::ElementPtr _sdf)

  *Load the road from SDF.*

### Additional Inherited Members

## 10.157.1 Detailed Description

for building a **Road** (p. 804) from SDF

## 10.157.2 Constructor & Destructor Documentation

#### 10.157.2.1 gazebo::physics::Road::Road ( BasePtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of this road object. |

#### 10.157.2.2 virtual gazebo::physics::Road::~Road ( ) [virtual]

Destructor.

## 10.157.3 Member Function Documentation

#### 10.157.3.1 virtual void gazebo::physics::Road::Init ( ) [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 167).

#### 10.157.3.2 void gazebo::physics::Road::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the road from SDF.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::Base** (p. 168).

The documentation for this class was generated from the following file:

- **Road.hh**

## 10.158 gazebo::rendering::Road2d Class Reference

`#include <Road2d.hh>`

### Public Member Functions

- **Road2d** ()
  *Constructor.*
- virtual ~**Road2d** ()
  *Destructor.*
- void **Load** (**VisualPtr** _parent)
  *Load the visual using a parent visual.*

### 10.158.1 Constructor & Destructor Documentation

#### 10.158.1.1 gazebo::rendering::Road2d::Road2d ( )

Constructor.

#### 10.158.1.2 virtual gazebo::rendering::Road2d::~Road2d ( ) [virtual]

Destructor.

### 10.158.2 Member Function Documentation

#### 10.158.2.1 void gazebo::rendering::Road2d::Load ( VisualPtr *_parent* )

Load the visual using a parent visual.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Pointer to the parent visual. |

The documentation for this class was generated from the following file:

- **Road2d.hh**

## 10.159 gazebo::math::RotationSpline Class Reference

**Spline** (p. 993) for rotations.

`#include <math/gzmath.hh>`

### Public Member Functions

- **RotationSpline** ()

  *Constructor. Sets the autoCalc to true.*
- ∼**RotationSpline** ()

  *Destructor. Nothing is done.*
- void **AddPoint** (const **Quaternion** &_p)

  *Adds a control point to the end of the spline.*
- void **Clear** ()

  *Clears all the points in the spline.*
- unsigned int **GetNumPoints** () const

  *Gets the number of control points in the spline.*
- const **Quaternion** & **GetPoint** (unsigned int _index) const

  *Gets the detail of one of the control points of the spline.*
- **Quaternion Interpolate** (double _t, bool _useShortestPath=true)

  *Returns an interpolated point based on a parametric value over the whole series.*
- **Quaternion Interpolate** (unsigned int _fromIndex, double _t, bool _useShortestPath=true)

  *Interpolates a single segment of the spline given a parametric value.*
- void **RecalcTangents** ()

  *Recalculates the tangents associated with this spline.*
- void **SetAutoCalculate** (bool _autoCalc)

  *Tells the spline whether it should automatically calculate tangents on demand as points are added.*
- void **UpdatePoint** (unsigned int _index, const **Quaternion** &_value)

  *Updates a single point in the spline.*

### Protected Attributes

- bool **autoCalc**

  *Automatic recalcultation of tangeants when control points are updated.*
- std::vector< **Quaternion** > **points**

  *the control points*
- std::vector< **Quaternion** > **tangents**

  *the tangents*

## 10.159.1 Detailed Description

**Spline** (p. 993) for rotations.

## 10.159.2 Constructor & Destructor Documentation

### 10.159.2.1 gazebo::math::RotationSpline::RotationSpline ( )

Constructor. Sets the autoCalc to true.

### 10.159.2.2 gazebo::math::RotationSpline::∼RotationSpline ( )

Destructor. Nothing is done.

### 10.159.3 Member Function Documentation

#### 10.159.3.1 void gazebo::math::RotationSpline::AddPoint ( const Quaternion & _p )

Adds a control point to the end of the spline.

**Parameters**

| | | |
|---|---|---|
| in | *_p* | control point |

#### 10.159.3.2 void gazebo::math::RotationSpline::Clear ( )

Clears all the points in the spline.

#### 10.159.3.3 unsigned int gazebo::math::RotationSpline::GetNumPoints ( ) const

Gets the number of control points in the spline.

**Returns**

the count

#### 10.159.3.4 const Quaternion& gazebo::math::RotationSpline::GetPoint ( unsigned int _index ) const

Gets the detail of one of the control points of the spline.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | the index of the control point. |

**Remarks**

This point must already exist in the spline.

**Returns**

a quaternion (out of bound index result in assertion)

#### 10.159.3.5 Quaternion gazebo::math::RotationSpline::Interpolate ( double _t, bool _useShortestPath = true )

Returns an interpolated point based on a parametric value over the whole series.

**Remarks**

Given a t value between 0 and 1 representing the parametric distance along the whole length of the spline, this method returns an interpolated point.

**Parameters**

| | | |
|---|---|---|
| in | *_t* | Parametric value. |
| in | *_useShortestPath* | Defines if rotation should take the shortest possible path |

**Returns**

the rotation

#### 10.159.3.6 Quaternion gazebo::math::RotationSpline::Interpolate ( unsigned int *_fromIndex,* double *_t,* bool *_useShortestPath =* `true` )

Interpolates a single segment of the spline given a parametric value.

**Parameters**

| | | |
|---|---|---|
| in | *_fromIndex* | The point index to treat as t = 0. _fromIndex + 1 is deemed to be t = 1 |
| in | *_t* | Parametric value |
| in | *_useShortest-Path* | Defines if rotation should take the shortest possible path |

**Returns**

the rotation

#### 10.159.3.7 void gazebo::math::RotationSpline::RecalcTangents ( )

Recalculates the tangents associated with this spline.

**Remarks**

If you tell the spline not to update on demand by calling setAutoCalculate(false) then you must call this after completing your updates to the spline points.

#### 10.159.3.8 void gazebo::math::RotationSpline::SetAutoCalculate ( bool *_autoCalc* )

Tells the spline whether it should automatically calculate tangents on demand as points are added.

**Remarks**

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the recalcTangents method when you are done.

**Parameters**

| | | |
|---|---|---|
| in | *_autoCalc* | If true, tangents are calculated for you whenever a point changes. If false, you must call reclacTangents to recalculate them when it best suits. |

#### 10.159.3.9 void gazebo::math::RotationSpline::UpdatePoint ( unsigned int *_index,* const Quaternion & *_value* )

Updates a single point in the spline.

**Remarks**

This point must already exist in the spline.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | index |
| in | *_value* | the new control point value |

### 10.159.4 Member Data Documentation

#### 10.159.4.1 bool gazebo::math::RotationSpline::autoCalc [protected]

Automatic recalcultation of tangeants when control points are updated.

#### 10.159.4.2 std::vector<Quaternion> gazebo::math::RotationSpline::points [protected]

the control points

#### 10.159.4.3 std::vector<Quaternion> gazebo::math::RotationSpline::tangents [protected]

the tangents

The documentation for this class was generated from the following file:

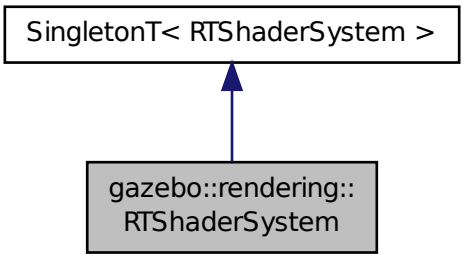
- **RotationSpline.hh**

## 10.160 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 130)'s Run-Time Shader system.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RTShaderSystem:



### Public Types

- enum **LightingModel** { **SSLM_PerVertexLighting**, **SSLM_PerPixelLighting**, **SSLM_NormalMapLighting-TangentSpace**, **SSLM_NormalMapLightingObjectSpace** }

### Public Member Functions

- void **AddScene** (**ScenePtr** _scene)
  *Add a scene manager.*
- void **ApplyShadows** (**ScenePtr** _scene)
  *Apply shadows to a scene.*
- void **AttachEntity** (**Visual** ∗vis)
  *Set an Ogre::Entity to use RT shaders.*
- void **Clear** ()
  *Clear the shader system.*
- void **DetachEntity** (**Visual** ∗_vis)
  *Remove and entity.*
- void **Fini** ()
  *Finalize the shader system.*
- void **GenerateShaders** (**Visual** ∗_vis)
  *Generate shaders for an entity.*
- Ogre::PSSMShadowCameraSetup ∗ **GetPSSMShadowCameraSetup** () const
  *Get the **Ogre** (p. 130) PSSM Shadows camera setup.*
- void **Init** ()
  *Init the run time shader system.*
- void **RemoveScene** (**ScenePtr** _scene)
  *Remove a scene.*
- void **RemoveShadows** (**ScenePtr** _scene)
  *Remove shadows from a scene.*
- void **SetPerPixelLighting** (bool _set)
  *Set the lighting model to per pixel or per vertex.*
- void **UpdateShaders** ()
  *Update the shaders. This should not be called frequently.*

### Static Public Member Functions

- static void **AttachViewport** (Ogre::Viewport ∗_viewport, **ScenePtr** _scene)
  *Set a viewport to use shaders.*
- static void **DetachViewport** (Ogre::Viewport ∗_viewport, **ScenePtr** _scene)
  *Set a viewport to not use shaders.*

### Additional Inherited Members

#### 10.160.1 Detailed Description

Implements **Ogre** (p. 130)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

### 10.160.2 Member Enumeration Documentation

#### 10.160.2.1 enum gazebo::rendering::RTShaderSystem::LightingModel

The type of lighting.

**Enumerator**

***SSLM_PerVertexLighting*** Per-Vertex lighting: best performance.

***SSLM_PerPixelLighting*** Per-Pixel lighting: best look.

***SSLM_NormalMapLightingTangentSpace*** Normal Map lighting: lighting calculations have been stored in a light map (texture) using tangent space.

***SSLM_NormalMapLightingObjectSpace*** Normal Map lighting: lighting calculations have been stored in a light map (texture) using object space.

### 10.160.3 Member Function Documentation

#### 10.160.3.1 void gazebo::rendering::RTShaderSystem::AddScene ( ScenePtr *_scene* )

Add a scene manager.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | The scene to process |

#### 10.160.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows ( ScenePtr *_scene* )

Apply shadows to a scene.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | The scene to receive shadows. |

#### 10.160.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity ( Visual * *vis* )

Set an Ogre::Entity to use RT shaders.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | **Visual** (p. 1121) that will use the **RTShaderSystem** (p. 810). |

#### 10.160.3.4 static void gazebo::rendering::RTShaderSystem::AttachViewport ( Ogre::Viewport * *_viewport,* ScenePtr *_scene* ) [static]

Set a viewport to use shaders.

**Parameters**

| | | |
|---|---|---|
| in | *_viewport* | The viewport to add. |
| in | *_scene* | The scene that the viewport uses. |

#### 10.160.3.5 void gazebo::rendering::RTShaderSystem::Clear ( )

Clear the shader system.

#### 10.160.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity ( Visual * _vis )

Remove and entity.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | Remove this visual. |

#### 10.160.3.7 static void gazebo::rendering::RTShaderSystem::DetachViewport ( Ogre::Viewport * _viewport, ScenePtr _scene ) [static]

Set a viewport to not use shaders.

**Parameters**

| | | |
|---|---|---|
| in | *_viewport* | The viewport to remove. |
| in | *_scene* | The scene that the viewport uses. |

#### 10.160.3.8 void gazebo::rendering::RTShaderSystem::Fini ( )

Finalize the shader system.

#### 10.160.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders ( Visual * _vis )

Generate shaders for an entity.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | The visual to generate shaders for. |

#### 10.160.3.10 Ogre::PSSMShadowCameraSetup* gazebo::rendering::RTShaderSystem::GetPSSMShadowCameraSetup ( ) const

Get the **Ogre** (p. 130) PSSM Shadows camera setup.

**Returns**

The **Ogre** (p. 130) PSSM Shadows camera setup.

#### 10.160.3.11 void gazebo::rendering::RTShaderSystem::Init ( )

Init the run time shader system.

#### 10.160.3.12 void gazebo::rendering::RTShaderSystem::RemoveScene ( ScenePtr _scene )

Remove a scene.

**Parameters**

| | | |
|---|---|---|
| in | *The* | scene to remove |

#### 10.160.3.13 void gazebo::rendering::RTShaderSystem::RemoveShadows ( ScenePtr _scene )

Remove shadows from a scene.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | The scene to remove shadows from. |

#### 10.160.3.14 void gazebo::rendering::RTShaderSystem::SetPerPixelLighting ( bool _set )

Set the lighting model to per pixel or per vertex.

**Parameters**

| | | |
|---|---|---|
| in | *_set* | True means to use per-pixel shaders. |

#### 10.160.3.15 void gazebo::rendering::RTShaderSystem::UpdateShaders ( )

Update the shaders. This should not be called frequently.

The documentation for this class was generated from the following file:

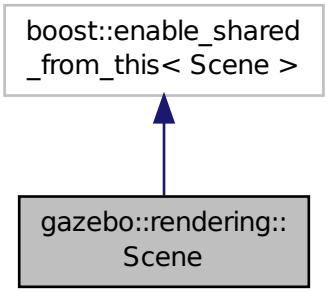
- **RTShaderSystem.hh**

## 10.161 gazebo::rendering::Scene Class Reference

Representation of an entire scene graph.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Scene:




## Public Types

- enum **SkyXMode** { **GZ_SKYX_ALL** = 0x0FFFFFFF, **GZ_SKYX_CLOUDS** = 0x0000001, **GZ_SKYX_MOON** = 0x0000002, **GZ_SKYX_NONE** = 0 }

## Public Member Functions

- **Scene** (const std::string &_name, bool _enableVisualizations=false, bool _isServer=false)

  *Constructor.*
- virtual ~**Scene** ()

  *Destructor.*
- void **AddVisual** (**VisualPtr** _vis)

  *Add a visual to the scene.*
- void **Clear** ()

  *Clear **rendering::Scene** (p. 814).*
- **VisualPtr CloneVisual** (const std::string &_visualName, const std::string &_newName) **GAZEBO_DEPRECATED**(2.0)

  *Deprecated.*
- **CameraPtr CreateCamera** (const std::string &_name, bool _autoRender=true)

  *Create a camera.*
- **DepthCameraPtr CreateDepthCamera** (const std::string &_name, bool _autoRender=true)

  *Create depth camera.*
- **GpuLaserPtr CreateGpuLaser** (const std::string &_name, bool _autoRender=true)

  *Create laser that generates data from rendering.*
- void **CreateGrid** (uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)

  *Create a square grid of cells.*
- **UserCameraPtr CreateUserCamera** (const std::string &_name)

  *Create a user camera.*
- void **DrawLine** (const **math::Vector3** &_start, const **math::Vector3** &_end, const std::string &_name)

*Draw a named line.*

- **common::Color GetAmbientColor** () const

  *Get the ambient color.*
- **common::Color GetBackgroundColor** () const

  *Get the background color.*
- **CameraPtr GetCamera** (uint32_t _index) const

  *Get a camera based on an index.*
- **CameraPtr GetCamera** (const std::string &_name) const

  *Get a camera by name.*
- uint32_t **GetCameraCount** () const

  *Get the number of cameras in this scene.*
- bool **GetFirstContact** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos, **math::Vector3** &_position)

  *Get the world pos of a the first contact at a pixel location.*
- **Grid** ∗ **GetGrid** (uint32_t _index) const

  *Get a grid based on an index.*
- uint32_t **GetGridCount** () const

  *Get the number of grids.*
- double **GetHeightBelowPoint** (const **math::Vector3** &_pt)

  *Get the Z-value of the first object below the given point.*
- **Heightmap** ∗ **GetHeightmap** () const

  *Get a pointer to the heightmap.*
- uint32_t **GetId** () const

  *Get the scene ID.*
- std::string **GetIdString** () const

  *Get the scene Id as a string.*
- bool **GetInitialized** () const

  *Return true if the **Scene** (p. 814) has been initialized.*
- **LightPtr GetLight** (const std::string &_name) const

  *Get a light by name.*
- **LightPtr GetLight** (uint32_t _index) const

  *Get a light based on an index.*
- uint32_t **GetLightCount** () const

  *Get the count of the lights.*
- Ogre::SceneManager ∗ **GetManager** () const

  *Get the OGRE scene manager.*
- **VisualPtr GetModelVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)

  *Get a model's visual at a mouse position.*
- std::string **GetName** () const

  *Get the name of the scene.*
- **VisualPtr GetSelectedVisual** () const

  *Get the currently selected visual.*
- bool **GetShadowsEnabled** () const

  *Get whether shadows are on or off.*
- bool **GetShowClouds** () const

  *Get whether or not clouds are displayed.*
- **common::Time GetSimTime** () const

  *Get the scene simulation time.*

- **UserCameraPtr GetUserCamera** (uint32_t _index) const

  *Get a user camera by index.*
- uint32_t **GetUserCameraCount** () const

  *Get the number of user cameras in this scene.*
- **VisualPtr GetVisual** (const std::string &_name) const

  *Get a visual by name.*
- **VisualPtr GetVisual** (uint32_t _id) const

  *Get a visual by id.*
- **VisualPtr GetVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos, std::string &_mod)

  *Get an entity at a pixel location using a camera.*
- **VisualPtr GetVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)

  *Get a visual at a mouse position.*
- **VisualPtr GetVisualBelow** (const std::string &_visualName)

  *Get the closest visual below a given visual.*
- uint32_t **GetVisualCount** () const

  *Get the number of visuals.*
- void **GetVisualsBelowPoint** (const **math::Vector3** &_pt, std::vector< **VisualPtr** > &_visuals)

  *Get a visual directly below a point.*
- **VisualPtr GetWorldVisual** () const

  *Get the top level world visual.*
- void **Init** ()

  *Init **rendering::Scene*** (p. *814*).
- void **Load** (sdf::ElementPtr _scene)

  *Load the scene from a set of parameters.*
- void **Load** ()

  *Load the scene with default parameters.*
- void **PreRender** ()

  *Process all received messages.*
- void **PrintSceneGraph** ()

  *Print the scene graph to std_out.*
- void **RemoveCamera** (const std::string &_name)

  *Remove a camera from the scene.*
- void **RemoveProjectors** ()

  *Remove all projectors.*
- void **RemoveVisual** (**VisualPtr** _vis)

  *Remove a visual from the scene.*
- void **SelectVisual** (const std::string &_name, const std::string &_mode)

  *Select a visual by name.*
- void **SetAmbientColor** (const **common::Color** &_color)

  *Set the ambient color.*
- void **SetBackgroundColor** (const **common::Color** &_color)

  *Set the background color.*
- void **SetFog** (const std::string &_type, const **common::Color** &_color, double _density, double _start, double _end)

  *Set the fog parameters.*
- void **SetGrid** (bool _enabled)

  *Set the grid on or off.*

- void **SetShadowsEnabled** (bool _value)

  *Set whether shadows are on or off.*
- void **SetSkyXMode** (unsigned int _mode)

  *Set **SkyX** (p. 130) mode to enable/disable skyx components such as clouds and moon.*
- void **SetTransparent** (bool _show)

  *Enable or disable transparency for all visuals.*
- void **SetVisible** (const std::string &_name, bool _visible)

  *Hide or show a visual.*
- void **SetWireframe** (bool _show)

  *Enable or disable wireframe for all visuals.*
- void **ShowClouds** (bool _show)

  *Display clouds in the sky.*
- void **ShowCollisions** (bool _show)

  *Enable or disable collision visualization.*
- void **ShowCOMs** (bool _show)

  *Enable or disable center of mass visualization.*
- void **ShowContacts** (bool _show)

  *Enable or disable contact visualization.*
- void **ShowJoints** (bool _show)

  *Enable or disable joint visualization.*
- void **SnapVisualToNearestBelow** (const std::string &_visualName)

  *Move the visual to be ontop of the nearest visual below it.*
- std::string **StripSceneName** (const std::string &_name) const

  *Remove the name of scene from a string.*

### Public Attributes

- SkyX::SkyX * **skyx**

  *Pointer to the sky.*

## 10.161.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

## 10.161.2 Member Enumeration Documentation

### 10.161.2.1 enum gazebo::rendering::Scene::SkyXMode

**Enumerator**

***GZ_SKYX_ALL***

***GZ_SKYX_CLOUDS***

***GZ_SKYX_MOON***

***GZ_SKYX_NONE***

### 10.161.3 Constructor & Destructor Documentation

#### 10.161.3.1 gazebo::rendering::Scene::Scene ( const std::string & *_name*, bool *_enableVisualizations* = `false`, bool *_isServer* = `false` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the scene. |
| in | *_enableVisualizations* | True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation. |

#### 10.161.3.2 virtual gazebo::rendering::Scene::~Scene ( ) [virtual]

Destructor.

### 10.161.4 Member Function Documentation

#### 10.161.4.1 void gazebo::rendering::Scene::AddVisual ( VisualPtr *_vis* )

Add a visual to the scene.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | **Visual** (p. 1121) to add. |

#### 10.161.4.2 void gazebo::rendering::Scene::Clear ( )

Clear **rendering::Scene** (p. 814).

#### 10.161.4.3 VisualPtr gazebo::rendering::Scene::CloneVisual ( const std::string & *_visualName*, const std::string & *_newName* )

Deprecated.

#### 10.161.4.4 CameraPtr gazebo::rendering::Scene::CreateCamera ( const std::string & *_name*, bool *_autoRender* = `true` )

Create a camera.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the new camera. |
| in | *_autoRender* | True to allow Gazebo to automatically render the camera. This should almost always be true. |

**Returns**

Pointer to the new camera.

#### 10.161.4.5 DepthCameraPtr gazebo::rendering::Scene::CreateDepthCamera ( const std::string & *_name,* bool *_autoRender* = `true` )

Create depth camera.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the new camera. |
| in | *_autoRender* | True to allow Gazebo to automatically render the camera. This should almost always be true. |

**Returns**

Pointer to the new camera.

#### 10.161.4.6 GpuLaserPtr gazebo::rendering::Scene::CreateGpuLaser ( const std::string & *_name,* bool *_autoRender* = `true` )

Create laser that generates data from rendering.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the new laser. |
| in | *_autoRender* | True to allow Gazebo to automatically render the camera. This should almost always be true. |

**Returns**

Pointer to the new laser.

#### 10.161.4.7 void gazebo::rendering::Scene::CreateGrid ( uint32_t *_cellCount,* float *_cellLength,* float *_lineWidth,* const common::Color & *_color* )

Create a square grid of cells.

**Parameters**

| | | |
|---|---|---|
| in | *_cellCount* | Number of grid cells in one direction. |
| in | *_cellLength* | Length of one grid cell. |
| in | *_lineWidth* | Width of the grid lines. |
| in | *_color* | Color of the grid lines. |

#### 10.161.4.8 UserCameraPtr gazebo::rendering::Scene::CreateUserCamera ( const std::string & *_name* )

Create a user camera.

A user camera is one design for use with a GUI.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the **UserCamera** (p. 1066). |

**Returns**

A pointer to the new **UserCamera** (p. 1066).

#### 10.161.4.9 void gazebo::rendering::Scene::DrawLine ( const math::Vector3 & _start, const math::Vector3 & _end, const std::string & _name )

Draw a named line.

**Parameters**

| | | |
|---|---|---|
| in | *_start* | Start position of the line. |
| in | *_end* | End position of the line. |
| in | *_name* | Name of the line. |

#### 10.161.4.10 common::Color gazebo::rendering::Scene::GetAmbientColor ( ) const

Get the ambient color.

**Returns**

The scene's ambient color.

#### 10.161.4.11 common::Color gazebo::rendering::Scene::GetBackgroundColor ( ) const

Get the background color.

**Returns**

The background color.

#### 10.161.4.12 CameraPtr gazebo::rendering::Scene::GetCamera ( uint32_t _index ) const

Get a camera based on an index.

Index must be between 0 and **Scene::GetCameraCount** (p. 822).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the camera to get. |

**Returns**

Pointer to the camera. Or NULL if the index is invalid.

#### 10.161.4.13 CameraPtr gazebo::rendering::Scene::GetCamera ( const std::string & _name ) const

Get a camera by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the camera. |

**Returns**

Pointer to the camera. Or NULL if the name is invalid.

#### 10.161.4.14 uint32_t gazebo::rendering::Scene::GetCameraCount ( ) const

Get the number of cameras in this scene.

**Returns**

Number of lasers.

#### 10.161.4.15 bool gazebo::rendering::Scene::GetFirstContact ( CameraPtr *_camera,* const math::Vector2i & *_mousePos,* math::Vector3 & *_position* )

Get the world pos of a the first contact at a pixel location.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | Pointer to the camera. |
| in | *_mousePos* | 2D position of the mouse in pixels. |
| out | *_position* | 3D position of the first contact point. |

**Returns**

True if a valid object was hit by the raycast.

#### 10.161.4.16 Grid* gazebo::rendering::Scene::GetGrid ( uint32_t *_index* ) const

Get a grid based on an index.

Index must be between 0 and **Scene::GetGridCount** (p. 822).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the grid. |

#### 10.161.4.17 uint32_t gazebo::rendering::Scene::GetGridCount ( ) const

Get the number of grids.

**Returns**

The number of grids.

#### 10.161.4.18 double gazebo::rendering::Scene::GetHeightBelowPoint ( const math::Vector3 & _pt )

Get the Z-value of the first object below the given point.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | Position to search below for a visual. |

**Returns**

The Z-value of the nearest visual below the point. Zero is returned if no visual is found.

#### 10.161.4.19 Heightmap* gazebo::rendering::Scene::GetHeightmap ( ) const

Get a pointer to the heightmap.

**Returns**

Pointer to the heightmap, NULL if no heightmap.

#### 10.161.4.20 uint32_t gazebo::rendering::Scene::GetId ( ) const

Get the scene ID.

**Returns**

The ID of the scene.

#### 10.161.4.21 std::string gazebo::rendering::Scene::GetIdString ( ) const

Get the scene Id as a string.

**Returns**

The ID as a string.

#### 10.161.4.22 bool gazebo::rendering::Scene::GetInitialized ( ) const

Return true if the **Scene** (p. 814) has been initialized.

#### 10.161.4.23 LightPtr gazebo::rendering::Scene::GetLight ( const std::string & _name ) const

Get a light by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the light to get. |

**Returns**

Pointer to the light, or NULL if the light was not found.

#### 10.161.4.24 LightPtr gazebo::rendering::Scene::GetLight ( uint32_t *_index* ) const

Get a light based on an index.

The index must be between 0 and **Scene::GetLightCount** (p. 824).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the light. |

**Returns**

Pointer to the **Light** (p. 535) or NULL if index was invalid.

#### 10.161.4.25 uint32_t gazebo::rendering::Scene::GetLightCount ( ) const

Get the count of the lights.

**Returns**

The number of lights.

#### 10.161.4.26 Ogre::SceneManager* gazebo::rendering::Scene::GetManager ( ) const

Get the OGRE scene manager.

**Returns**

Pointer to the **Ogre** (p. 130) SceneManager.

#### 10.161.4.27 VisualPtr gazebo::rendering::Scene::GetModelVisualAt ( CameraPtr *_camera,* const math::Vector2i & *_mousePos* )

Get a model's visual at a mouse position.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | Pointer to the camera used to project the mouse position. |
| in | *_mousePos* | The 2d position of the mouse in pixels. |

**Returns**

Pointer to the visual, NULL if none found.

#### 10.161.4.28 std::string gazebo::rendering::Scene::GetName ( ) const

Get the name of the scene.

**Returns**

Name of the scene.

#### 10.161.4.29 VisualPtr gazebo::rendering::Scene::GetSelectedVisual ( ) const

Get the currently selected visual.

**Returns**

Pointer to the currently selected visual, or NULL if nothing is selected.

#### 10.161.4.30 bool gazebo::rendering::Scene::GetShadowsEnabled ( ) const

Get whether shadows are on or off.

**Returns**

True if shadows are enabled.

#### 10.161.4.31 bool gazebo::rendering::Scene::GetShowClouds ( ) const

Get whether or not clouds are displayed.

**Returns**

True if clouds are displayed.

#### 10.161.4.32 common::Time gazebo::rendering::Scene::GetSimTime ( ) const

Get the scene simulation time.

Note this is different from World::GetSimTime() because there is a lag between the time new poses are sent out by World and when they are received and applied by the **Scene** (p. 814).

**Returns**

The current simulation time in **Scene** (p. 814)

#### 10.161.4.33 UserCameraPtr gazebo::rendering::Scene::GetUserCamera ( uint32_t *_index* ) const

Get a user camera by index.

The index value must be between 0 and **Scene::GetUserCameraCount** (p. 826).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the **UserCamera** (p. 1066) to get. |

**Returns**

Pointer to the **UserCamera** (p. 1066), or NULL if the index was invalid.

#### 10.161.4.34 uint32_t gazebo::rendering::Scene::GetUserCameraCount ( ) const

Get the number of user cameras in this scene.

**Returns**

The number of user cameras.

#### 10.161.4.35 VisualPtr gazebo::rendering::Scene::GetVisual ( const std::string & *_name* ) const

Get a visual by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual to retrieve. |

**Returns**

Pointer to the visual, NULL if not found.

#### 10.161.4.36 VisualPtr gazebo::rendering::Scene::GetVisual ( uint32_t *_id* ) const

Get a visual by id.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the visual to retrieve. |

**Returns**

Pointer to the visual, NULL if not found.

#### 10.161.4.37 VisualPtr gazebo::rendering::Scene::GetVisualAt ( CameraPtr *_camera,* const math::Vector2i & *_mousePos,* std::string & *_mod* )

Get an entity at a pixel location using a camera.

Used for mouse picking.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | The ogre camera, used to do mouse picking |
| in | *_mousePos* | The position of the mouse in screen coordinates |
| out | *_mod* | Used for object manipulation |

**Returns**

The selected entity, or NULL

#### 10.161.4.38 VisualPtr gazebo::rendering::Scene::GetVisualAt ( CameraPtr *_camera,* const math::Vector2i & *_mousePos* )

Get a visual at a mouse position.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | Pointer to the camera used to project the mouse position. |
| in | *_mousePos* | The 2d position of the mouse in pixels. |

**Returns**

Pointer to the visual, NULL if none found.

#### 10.161.4.39 VisualPtr gazebo::rendering::Scene::GetVisualBelow ( const std::string & *_visualName* )

Get the closest visual below a given visual.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to search below. |

**Returns**

Pointer to the visual below, or NULL if no visual.

#### 10.161.4.40 uint32_t gazebo::rendering::Scene::GetVisualCount ( ) const

Get the number of visuals.

**Returns**

The number of visuals in the **Scene** (p. 814).

#### 10.161.4.41 void gazebo::rendering::Scene::GetVisualsBelowPoint ( const math::Vector3 & *_pt,* std::vector< VisualPtr > & *_visuals* )

Get a visual directly below a point.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | 3D point to get the visual below. |
| out | *_visuals* | The visuals below the point order in proximity. |

#### 10.161.4.42 VisualPtr gazebo::rendering::Scene::GetWorldVisual ( ) const

Get the top level world visual.

**Returns**

Pointer to the world visual.

#### 10.161.4.43 void gazebo::rendering::Scene::Init ( )

Init **rendering::Scene** (p. 814).

#### 10.161.4.44 void gazebo::rendering::Scene::Load ( sdf::ElementPtr _scene )

Load the scene from a set of parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | SDF scene element to load. |

#### 10.161.4.45 void gazebo::rendering::Scene::Load ( )

Load the scene with default parameters.

#### 10.161.4.46 void gazebo::rendering::Scene::PreRender ( )

Process all received messages.

#### 10.161.4.47 void gazebo::rendering::Scene::PrintSceneGraph ( )

Print the scene graph to std_out.

#### 10.161.4.48 void gazebo::rendering::Scene::RemoveCamera ( const std::string & _name )

Remove a camera from the scene.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the camera. |

#### 10.161.4.49 void gazebo::rendering::Scene::RemoveProjectors ( )

Remove all projectors.

#### 10.161.4.50 void gazebo::rendering::Scene::RemoveVisual ( VisualPtr _vis )

Remove a visual from the scene.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | **Visual** (p. 1121) to remove. |

#### 10.161.4.51 void gazebo::rendering::Scene::SelectVisual ( const std::string & *_name,* const std::string & *_mode* )

Select a visual by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual to select. |
| in | *_mode* | Selection mode (normal, or move). |

#### 10.161.4.52 void gazebo::rendering::Scene::SetAmbientColor ( const common::Color & *_color* )

Set the ambient color.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | The ambient color to use. |

#### 10.161.4.53 void gazebo::rendering::Scene::SetBackgroundColor ( const common::Color & *_color* )

Set the background color.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | The background color. |

#### 10.161.4.54 void gazebo::rendering::Scene::SetFog ( const std::string & *_type,* const common::Color & *_color,* double *_density,* double *_start,* double *_end* )

Set the fog parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | Type of fog: "linear", "exp", or "exp2". |
| in | *_color* | Color of the fog. |
| in | *_density* | Fog density. |
| in | *_start* | Distance from camera to start the fog. |
| in | *_end* | Distance from camera at which the fog is at max density. |

#### 10.161.4.55 void gazebo::rendering::Scene::SetGrid ( bool *_enabled* )

Set the grid on or off.

**Parameters**

| | | |
|---|---|---|
| in | *_enabled* | Set to true to turn on the grid |

#### 10.161.4.56 void gazebo::rendering::Scene::SetShadowsEnabled ( bool *_value* )

Set whether shadows are on or off.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | True to enable shadows, False to disable |

#### 10.161.4.57 void gazebo::rendering::Scene::SetSkyXMode ( unsigned int *_mode* )

Set **SkyX** (p. 130) mode to enable/disable skyx components such as clouds and moon.

**Parameters**

| | | |
|---|---|---|
| in | *_mode* | **SkyX** (p. 130) mode bitmask. |

**See Also**

**Scene::SkyXMode** (p. 818)

#### 10.161.4.58 void gazebo::rendering::Scene::SetTransparent ( bool *_show* )

Enable or disable transparency for all visuals.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to enable transparency for all visuals. |

#### 10.161.4.59 void gazebo::rendering::Scene::SetVisible ( const std::string & *_name,* bool *_visible* )

Hide or show a visual.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual to change. |
| in | *_visible* | True to make visual visible, False to make it invisible. |

#### 10.161.4.60 void gazebo::rendering::Scene::SetWireframe ( bool *_show* )

Enable or disable wireframe for all visuals.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to enable wireframe for all visuals. |

#### 10.161.4.61 void gazebo::rendering::Scene::ShowClouds ( bool _show )

Display clouds in the sky.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to display clouds. |

#### 10.161.4.62 void gazebo::rendering::Scene::ShowCollisions ( bool _show )

Enable or disable collision visualization.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to enable collision visualization. |

#### 10.161.4.63 void gazebo::rendering::Scene::ShowCOMs ( bool _show )

Enable or disable center of mass visualization.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to enable center of mass visualization. |

#### 10.161.4.64 void gazebo::rendering::Scene::ShowContacts ( bool _show )

Enable or disable contact visualization.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to enable contact visualization. |

#### 10.161.4.65 void gazebo::rendering::Scene::ShowJoints ( bool _show )

Enable or disable joint visualization.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to enable joint visualization. |

#### 10.161.4.66 void gazebo::rendering::Scene::SnapVisualToNearestBelow ( const std::string & _visualName )

Move the visual to be ontop of the nearest visual below it.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to move. |

#### 10.161.4.67 std::string gazebo::rendering::Scene::StripSceneName ( const std::string & _name ) const

Remove the name of scene from a string.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name to string the scene name from. |

**Returns**

The stripped name.

### 10.161.5 Member Data Documentation

#### 10.161.5.1 SkyX::SkyX* gazebo::rendering::Scene::skyx

Pointer to the sky.

The documentation for this class was generated from the following file:

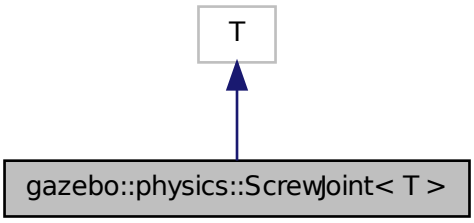
- **Scene.hh**

## 10.162 gazebo::physics::ScrewJoint< T > Class Template Reference

A screw joint, which has both prismatic and rotational DOFs.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::ScrewJoint< T >:




### Public Member Functions

- **ScrewJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**ScrewJoint** ()

  *Destructor.*

- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor.*
- virtual unsigned int **GetAngleCount** () const

- virtual double **GetThreadPitch** (unsigned int _index)=0

  *Get screw joint thread pitch.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **ScrewJoint** (p. 832).*
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)

  *Set the anchor.*
- virtual void **SetThreadPitch** (int _index, double _threadPitch)=0

  *Set screw joint thread pitch.*

## Protected Attributes

- **math::Vector3 fakeAnchor**

  *The anchor value is not used internally.*
- double **threadPitch**

  *Pitch of the thread.*

### 10.162.1 Detailed Description

**template<class T>class gazebo::physics::ScrewJoint< T >**

A screw joint, which has both prismatic and rotational DOFs.

### 10.162.2 Constructor & Destructor Documentation

#### 10.162.2.1 template<class T> gazebo::physics::ScrewJoint< T >::ScrewJoint ( BasePtr *_parent* ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the joint. |

#### 10.162.2.2 template<class T> virtual gazebo::physics::ScrewJoint< T >::~ScrewJoint ( ) [inline], [virtual]

Destructor.

### 10.162.3 Member Function Documentation

#### 10.162.3.1 template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor ( int _index ) const [virtual]

Get the anchor.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. Not Used. |

**Returns**

Anchor for the joint.

#### 10.162.3.2 template<class T> virtual unsigned int gazebo::physics::ScrewJoint< T >::GetAngleCount ( ) const [inline], [virtual]

#### 10.162.3.3 template<class T> virtual double gazebo::physics::ScrewJoint< T >::GetThreadPitch ( unsigned int _index ) [pure virtual]

Get screw joint thread pitch.

This must be implemented in a child class

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

_threadPitch Thread pitch value.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 933), and **gazebo::physics::DARTScrewJoint** (p. 342).

#### 10.162.3.4 template<class T> virtual void gazebo::physics::ScrewJoint< T >::Load ( sdf::ElementPtr _sdf ) [inline], [virtual]

Load a **ScrewJoint** (p. 832).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF value to load from |

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 934), and **gazebo::physics::DARTScrewJoint** (p. 343).

#### 10.162.3.5 template<class T > void gazebo::physics::ScrewJoint< T >::SetAnchor ( int _index, const math::Vector3 & _anchor ) [virtual]

Set the anchor.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. Not Used. |
| in | *_anchor* | Anchor value for the joint. |

#### 10.162.3.6 template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetThreadPitch ( int *_index*, double *_threadPitch* ) [pure virtual]

Set screw joint thread pitch.

This must be implemented in a child class

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_threadPitch* | Thread pitch value. |

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 936), and **gazebo::physics::DARTScrewJoint** (p. 344).

### 10.162.4 Member Data Documentation

#### 10.162.4.1 template<class T> math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor [protected]

The anchor value is not used internally.

#### 10.162.4.2 template<class T> double gazebo::physics::ScrewJoint< T >::threadPitch [protected]

Pitch of the thread.

Referenced by gazebo::physics::ScrewJoint< SimbodyJoint >::Load().

The documentation for this class was generated from the following file:

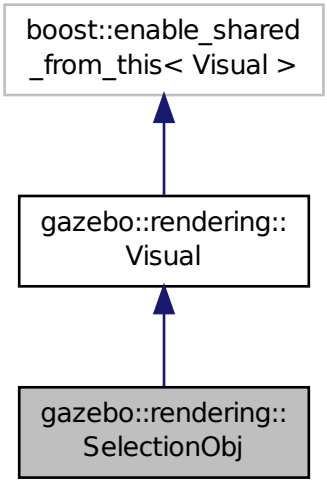
- **ScrewJoint.hh**

## 10.163 gazebo::rendering::SelectionObj Class Reference

Interactive selection object for models and links.

`#include <SelectionObj.hh>`

Inheritance diagram for gazebo::rendering::SelectionObj:




## Public Types

- enum **SelectionMode** {
  **SELECTION_NONE** = 0, **TRANS**, **ROT**, **SCALE**,
  **TRANS_X**, **TRANS_Y**, **TRANS_Z**, **ROT_X**,
  **ROT_Y**, **ROT_Z**, **SCALE_X**, **SCALE_Y**,
  **SCALE_Z** }

## Public Member Functions

- **SelectionObj** (const std::string &_name, **VisualPtr** _vis)

  *Constructor.*
- virtual ~**SelectionObj** ()

  *Deconstructor.*
- void **Attach** (**rendering::VisualPtr** _vis)

  *Attach the selection object to the given visual.*
- void **Detach** ()

  *Detach the selection object from the current visual.*
- **SelectionMode GetMode** ()

  *Get the current selection mode.*
- **SelectionMode GetState** ()

  *Get the current selection state.*
- void **Load** ()

  *Load.*
- void **SetGlobal** (bool _global)

*Set selection object to ignore local transforms.*

- void **SetMode** (const std::string &_mode)

  *Set the manipulation mode.*

- void **SetMode** (**SelectionMode** _mode)

  *Set the selection mode.*

- void **SetState** (const std::string &_state)

  *Set state by highlighting the corresponding selection object visual.*

- void **SetState** (**SelectionMode** _state)

  *Set state by highlighting the corresponding selection object visual.*

- void **UpdateSize** ()

  *Update selection object size to match the parent visual.*

### Additional Inherited Members

#### 10.163.1 Detailed Description

Interactive selection object for models and links.

The documentation for this class was generated from the following file:

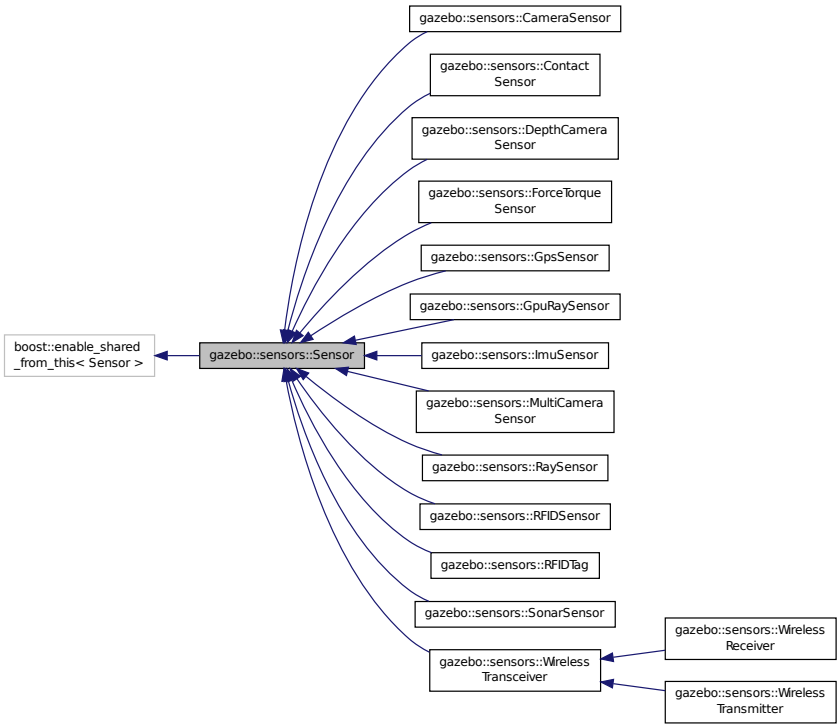
- **SelectionObj.hh**

## 10.164 gazebo::sensors::Sensor Class Reference

Base class for sensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::Sensor:




## Public Member Functions

- **Sensor** (**SensorCategory** _cat)

  *Constructor.*
- virtual ~**Sensor** ()

  *Destructor.*
- template<typename T >
  **event::ConnectionPtr ConnectUpdated** (T _subscriber)

  *Connect a signal that is triggered when the sensor is updated.*
- void **DisconnectUpdated** (**event::ConnectionPtr** &_c)

  *Disconnect from a the updated signal.*
- void **FillMsg** (msgs::Sensor &_msg)

  *fills a msgs::Sensor message.*
- virtual void **Fini** ()

  *Finalize the sensor.*
- **SensorCategory GetCategory** () const

  *Get the category of the sensor.*
- uint32_t **GetId** () const

  *Get the sensor's ID.*
- **common::Time GetLastMeasurementTime** ()

  *Return last measurement time.*

- **common::Time GetLastUpdateTime** ()

  *Return last update time.*
- std::string **GetName** () const

  *Get name.*
- uint32_t **GetParentId** () const

  *Get the sensor's parent's ID.*
- std::string **GetParentName** () const

  *Returns the name of the sensor parent.*
- virtual **math::Pose GetPose** () const

  *Get the current pose.*
- std::string **GetScopedName** () const

  *Get fully scoped name of the sensor.*
- virtual std::string **GetTopic** () const

  *Returns the topic name as set in SDF.*
- std::string **GetType** () const

  *Get sensor type.*
- double **GetUpdateRate** ()

  *Get the update rate of the sensor.*
- bool **GetVisualize** () const

  *Return true if user requests the sensor to be visualized via tag: <visualize>true</visualize> in SDF.*
- std::string **GetWorldName** () const

  *Returns the name of the world the sensor is in.*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  *Load the sensor with SDF parameters.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*
- void **ResetLastUpdateTime** ()

  *Reset the lastUpdateTime to zero.*
- virtual void **SetActive** (bool _value)

  *Set whether the sensor is active or not.*
- virtual void **SetParent** (const std::string &_name) **GAZEBO_DEPRECATED**(2.0)

  *Set the parent of the sensor.*
- void **SetParent** (const std::string &_name, uint32_t _id)

  *Set the sensor's parent.*
- void **SetUpdateRate** (double _hz)

  *Set the update rate of the sensor.*
- void **Update** (bool _force)

  *Update the sensor.*

## Protected Member Functions

- virtual void **UpdateImpl** (bool)

  *This gets overwritten by derived sensor types.*

### Protected Attributes

- bool **active**

  *True if sensor generation is active.*
- std::vector< **event::ConnectionPtr** > **connections**

  *All event connections.*
- **common::Time lastMeasurementTime**

  *Stores last time that a sensor measurement was generated; this value must be updated within each sensor's UpdateImpl.*
- **common::Time lastUpdateTime**

  *Time of the last update.*
- boost::mutex **mutexLastUpdateTime**

  *Mutex to protect resetting lastUpdateTime.*
- **transport::NodePtr node**

  *Node for communication.*
- uint32_t **parentId**

  *The sensor's parent ID.*
- std::string **parentName**

  *Name of the parent.*
- std::vector< **SensorPluginPtr** > **plugins**

  *All the plugins for the sensor.*
- **math::Pose pose**

  *Pose of the sensor.*
- **transport::SubscriberPtr poseSub**

  *Subscribe to pose updates.*
- **gazebo::rendering::ScenePtr scene**

  *Pointer to the Scene.*
- sdf::ElementPtr **sdf**

  *Pointer the the SDF element for the sensor.*
- **common::Time updatePeriod**

  *Desired time between updates, set indirectly by* ***Sensor::SetUpdateRate*** *(p. 846).*
- **gazebo::physics::WorldPtr world**

  *Pointer to the world.*

## 10.164.1 Detailed Description

Base class for sensors.

## 10.164.2 Constructor & Destructor Documentation

### 10.164.2.1 gazebo::sensors::Sensor::Sensor ( SensorCategory *_cat* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_class* | |

#### 10.164.2.2 virtual gazebo::sensors::Sensor::~Sensor ( ) [virtual]

Destructor.

### 10.164.3 Member Function Documentation

#### 10.164.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::Sensor::ConnectUpdated ( T _subscriber ) [inline]

Connect a signal that is triggered when the sensor is updated.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback that receives the signal. |

**Returns**

A pointer to the connection. This must be kept in scope.

**See Also**

**Sensor::DisconnectUpdated** (p. 841)

References gazebo::event::EventT< T >::Connect().

#### 10.164.3.2 void gazebo::sensors::Sensor::DisconnectUpdated ( event::ConnectionPtr & _c ) [inline]

Disconnect from a the updated signal.

**Parameters**

| | | |
|---|---|---|
| in | *_c* | The connection to disconnect |

**See Also**

**Sensor::ConnectUpdated** (p. 841)

References gazebo::event::EventT< T >::Disconnect().

#### 10.164.3.3 void gazebo::sensors::Sensor::FillMsg ( msgs::Sensor & _msg )

fills a msgs::Sensor message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | Message to fill. |

#### 10.164.3.4 virtual void gazebo::sensors::Sensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 661), **gazebo::sensors::ForceTorqueSensor** (p. 421), **gazebo::sensors::GpuRaySensor** (p. 442), **gazebo::sensors::CameraSensor** (p. 215), **gazebo::sensors::ContactSensor** (p. 269), **gazebo::sensors::RFIDSensor** (p. 797), **gazebo::sensors::RaySensor** (p. 781), **gazebo::sensors::DepthCameraSensor** (p. 363), **gazebo::sensors::RFIDTag** (p. 800), **gazebo::sensors::GpsSensor** (p. 427), **gazebo::sensors::SonarSensor** (p. 980), **gazebo::sensors::ImuSensor** (p. 481), **gazebo::sensors::WirelessTransceiver** (p. 1152), and **gazebo::sensors::WirelessReceiver** (p. 1149).

#### 10.164.3.5 SensorCategory gazebo::sensors::Sensor::GetCategory ( ) const

Get the category of the sensor.

**Returns**

The category of the sensor.

**See Also**

**SensorCategory** (p. 126)

#### 10.164.3.6 uint32_t gazebo::sensors::Sensor::GetId ( ) const

Get the sensor's ID.

**Returns**

The sensor's ID.

#### 10.164.3.7 common::Time gazebo::sensors::Sensor::GetLastMeasurementTime ( )

Return last measurement time.

**Returns**

Time of last measurement.

#### 10.164.3.8 common::Time gazebo::sensors::Sensor::GetLastUpdateTime ( )

Return last update time.

**Returns**

Time of last update.

#### 10.164.3.9 std::string gazebo::sensors::Sensor::GetName ( ) const

Get name.

**Returns**

Name of sensor.

#### 10.164.3.10 uint32_t gazebo::sensors::Sensor::GetParentId ( ) const

Get the sensor's parent's ID.

**Returns**

The sensor's parent's ID.

#### 10.164.3.11 std::string gazebo::sensors::Sensor::GetParentName ( ) const

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 846).

**Returns**

Name of Parent.

#### 10.164.3.12 virtual math::Pose gazebo::sensors::Sensor::GetPose ( ) const [virtual]

Get the current pose.

**Returns**

Current pose of the sensor.

#### 10.164.3.13 std::string gazebo::sensors::Sensor::GetScopedName ( ) const

Get fully scoped name of the sensor.

**Returns**

world_name::parent_name::sensor_name.

#### 10.164.3.14 virtual std::string gazebo::sensors::Sensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

**Returns**

Topic name.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 446), **gazebo::sensors::RaySensor** (p. 784), **gazebo::sensors::CameraSensor** (p. 215), **gazebo::sensors::SonarSensor** (p. 981), **gazebo::sensors::MultiCameraSensor** (p. 663), **gazebo::sensors::ForceTorqueSensor** (p. 421), and **gazebo::sensors::WirelessTransceiver** (p. 1152).

#### 10.164.3.15 std::string gazebo::sensors::Sensor::GetType ( ) const

Get sensor type.

**Returns**

Type of sensor.

#### 10.164.3.16 double gazebo::sensors::Sensor::GetUpdateRate ( )

Get the update rate of the sensor.

**Returns**

_hz update rate of sensor. Returns 0 if unthrottled.

#### 10.164.3.17 bool gazebo::sensors::Sensor::GetVisualize ( ) const

Return true if user requests the sensor to be visualized via tag: <visualize>true</visualize> in SDF.

**Returns**

True if visualized, false if not.

#### 10.164.3.18 std::string gazebo::sensors::Sensor::GetWorldName ( ) const

Returns the name of the world the sensor is in.

**Returns**

Name of the world.

#### 10.164.3.19 virtual void gazebo::sensors::Sensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 447), **gazebo::sensors::ContactSensor** (p. 271), **gazebo::sensors::RFIDSensor** (p. 797), **gazebo::sensors::RaySensor** (p. 785), **gazebo::sensors::CameraSensor** (p. 216), **gazebo::sensors::DepthCameraSensor** (p. 364), **gazebo::sensors::WirelessTransmitter** (p. 1156), **gazebo::sensors::RFIDTag** (p. 800), **gazebo::sensors::GpsSensor** (p. 428), **gazebo::sensors::SonarSensor** (p. 981), **gazebo::sensors::MultiCameraSensor** (p. 663), **gazebo::sensors::ImuSensor** (p. 482), **gazebo::sensors::WirelessTransceiver** (p. 1152), **gazebo::sensors::ForceTorqueSensor** (p. 421), and **gazebo::sensors::WirelessReceiver** (p. 1150).

#### 10.164.3.20 virtual bool gazebo::sensors::Sensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 447), **gazebo::sensors::RaySensor** (p. 785), **gazebo::sensors::ContactSensor** (p. 271), **gazebo::sensors::CameraSensor** (p. 216), **gazebo::sensors::MultiCameraSensor** (p. 663), **gazebo::sensors::SonarSensor** (p. 981), **gazebo::sensors::ImuSensor** (p. 482), and **gazebo::sensors::ForceTorqueSensor** (p. 422).

#### 10.164.3.21 virtual void gazebo::sensors::Sensor::Load ( const std::string & *_worldName,* sdf::ElementPtr *_sdf* ) [virtual]

Load the sensor with SDF parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF **Sensor** (p. 837) parameters. |
| in | *_worldName* | Name of world to load from. |

Reimplemented in **gazebo::sensors::ContactSensor** (p. 271), **gazebo::sensors::RFIDSensor** (p. 797), **gazebo::sensors::CameraSensor** (p. 216), and **gazebo::sensors::ImuSensor** (p. 483).

#### 10.164.3.22 virtual void gazebo::sensors::Sensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 448), **gazebo::sensors::ContactSensor** (p. 271), **gazebo::sensors::RFIDSensor** (p. 798), **gazebo::sensors::RaySensor** (p. 785), **gazebo::sensors::CameraSensor** (p. 216), **gazebo::sensors::DepthCameraSensor** (p. 364), **gazebo::sensors::WirelessTransmitter** (p. 1156), **gazebo::sensors::RFIDTag** (p. 800), **gazebo::sensors::GpsSensor** (p. 428), **gazebo::sensors::SonarSensor** (p. 981), **gazebo::sensors::MultiCameraSensor** (p. 664), **gazebo::sensors::ImuSensor** (p. 483), **gazebo::sensors::WirelessTransceiver** (p. 1152), **gazebo::sensors::ForceTorqueSensor** (p. 422), and **gazebo::sensors::WirelessReceiver** (p. 1150).

#### 10.164.3.23 void gazebo::sensors::Sensor::ResetLastUpdateTime ( )

Reset the lastUpdateTime to zero.

#### 10.164.3.24 virtual void gazebo::sensors::Sensor::SetActive ( bool *_value* ) [virtual]

Set whether the sensor is active or not.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | True if active, false if not. |

Reimplemented in **gazebo::sensors::DepthCameraSensor** (p. 364).

#### 10.164.3.25 virtual void gazebo::sensors::Sensor::SetParent ( const std::string & *_name* ) [virtual]

Set the parent of the sensor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the parent. |

#### 10.164.3.26 void gazebo::sensors::Sensor::SetParent ( const std::string & *_name*, uint32_t *_id* )

Set the sensor's parent.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The sensor's parent's name. |
| in | *_id* | The sensor's parent's ID. |

#### 10.164.3.27 void gazebo::sensors::Sensor::SetUpdateRate ( double *_hz* )

Set the update rate of the sensor.

**Parameters**

| | | |
|---|---|---|
| in | *_hz* | update rate of sensor. |

#### 10.164.3.28 void gazebo::sensors::Sensor::Update ( bool *_force* )

Update the sensor.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True to force update, false otherwise. |

#### 10.164.3.29 virtual void gazebo::sensors::Sensor::UpdateImpl ( bool ) [inline], [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 664), **gazebo::sensors::ForceTorqueSensor** (p. 422), **gazebo::sensors::GpuRaySensor** (p. 449), **gazebo::sensors::CameraSensor** (p. 217), **gazebo::sensors::-ContactSensor** (p. 272), **gazebo::sensors::RFIDSensor** (p. 798), **gazebo::sensors::RaySensor** (p. 785), **gazebo-::sensors::DepthCameraSensor** (p. 365), **gazebo::sensors::RFIDTag** (p. 800), **gazebo::sensors::GpsSensor**

(p. 428), **gazebo::sensors::SonarSensor** (p. 981), **gazebo::sensors::WirelessTransmitter** (p. 1156), and **gazebo::sensors::ImuSensor** (p. 483).

### 10.164.4 Member Data Documentation

#### 10.164.4.1 bool gazebo::sensors::Sensor::active [protected]

True if sensor generation is active.

#### 10.164.4.2 std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections [protected]

All event connections.

#### 10.164.4.3 common::Time gazebo::sensors::Sensor::lastMeasurementTime [protected]

Stores last time that a sensor measurement was generated; this value must be updated within each sensor's UpdateImpl.

#### 10.164.4.4 common::Time gazebo::sensors::Sensor::lastUpdateTime [protected]

Time of the last update.

#### 10.164.4.5 boost::mutex gazebo::sensors::Sensor::mutexLastUpdateTime [protected]

Mutex to protect resetting lastUpdateTime.

#### 10.164.4.6 transport::NodePtr gazebo::sensors::Sensor::node [protected]

Node for communication.

#### 10.164.4.7 uint32_t gazebo::sensors::Sensor::parentId [protected]

The sensor's parent ID.

#### 10.164.4.8 std::string gazebo::sensors::Sensor::parentName [protected]

Name of the parent.

#### 10.164.4.9 std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins [protected]

All the plugins for the sensor.

#### 10.164.4.10 math::Pose gazebo::sensors::Sensor::pose [protected]

Pose of the sensor.

#### 10.164.4.11 transport::SubscriberPtr gazebo::sensors::Sensor::poseSub [protected]

Subscribe to pose updates.

#### 10.164.4.12 gazebo::rendering::ScenePtr gazebo::sensors::Sensor::scene [protected]

Pointer to the Scene.

#### 10.164.4.13 sdf::ElementPtr gazebo::sensors::Sensor::sdf [protected]

Pointer the the SDF element for the sensor.

#### 10.164.4.14 common::Time gazebo::sensors::Sensor::updatePeriod [protected]

Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 846).

#### 10.164.4.15 gazebo::physics::WorldPtr gazebo::sensors::Sensor::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **Sensor.hh**

## 10.165 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

`#include <sensors/sensors.hh>`

### 10.165.1 Detailed Description

The sensor factory; the class is just for namespacing purposes.

The documentation for this class was generated from the following file:

- **SensorFactory.hh**

## 10.166 gazebo::sensors::SensorFactory Class Reference

`#include <SensorFactory.hh>`

### Static Public Member Functions

- static void **GetSensorTypes** (std::vector< std::string > &_types)

  *Get all the sensor types.*
- static **SensorPtr NewSensor** (const std::string &_className)

*Create a new instance of a sensor.*

- static void **RegisterAll** ()

  *Register all known sensors.*

- static void **RegisterSensor** (const std::string &_className, **SensorFactoryFn** _factoryfn)

  *Register a sensor class (called by sensor registration function).*

## 10.166.1 Member Function Documentation

### 10.166.1.1 static void gazebo::sensors::SensorFactory::GetSensorTypes ( std::vector< std::string > & *_types* ) [static]

Get all the sensor types.

**Parameters**

| | |
|---|---|
| *_types* | Vector of strings of the sensor types, populated by function |

### 10.166.1.2 static SensorPtr gazebo::sensors::SensorFactory::NewSensor ( const std::string & *_className* ) [static]

Create a new instance of a sensor.

Used by the world when reading the world file.

**Parameters**

| | | |
|---|---|---|
| in | *_className* | Name of sensor class |

**Returns**

Pointer to **Sensor** (p. 837)

### 10.166.1.3 static void gazebo::sensors::SensorFactory::RegisterAll ( ) [static]

Register all known sensors.

- **sensors::CameraSensor** (p. 213)
- **sensors::DepthCameraSensor** (p. 362)
- **sensors::GpuRaySensor** (p. 438)
- **sensors::RaySensor** (p. 779)
- **sensors::ContactSensor** (p. 267)
- **sensors::RFIDSensor** (p. 796)
- **sensors::RFIDTag** (p. 798)
- **sensors::WirelessTransmitter** (p. 1153)
- **sensors::WirelessReceiver** (p. 1147)

#### 10.166.1.4 static void gazebo::sensors::SensorFactory::RegisterSensor ( const std::string & *_className,* SensorFactoryFn *_factoryfn* ) [static]

Register a sensor class (called by sensor registration function).

**Parameters**

| | | |
|---|---|---|
| in | *_className* | Name of class of sensor to register. |
| in | *_factoryfn* | Function handle for registration. |

The documentation for this class was generated from the following file:

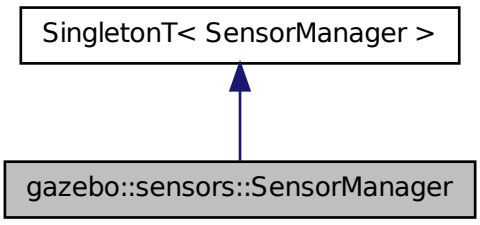
- **SensorFactory.hh**

## 10.167 gazebo::sensors::SensorManager Class Reference

Class to manage and update all sensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::SensorManager:




### Public Member Functions

- std::string **CreateSensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(2.0)

  *Deprecated.*
- std::string **CreateSensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  *Add a sensor from an SDF element.*
- void **Fini** ()

  *Finalize all the sensors.*
- **SensorPtr GetSensor** (const std::string &_name) const

  *Get a sensor.*
- **Sensor_V GetSensors** () const

  *Get all the sensors.*

- void **GetSensorTypes** (std::vector< std::string > &_types) const

  *Get all the sensor types.*
- void **Init** ()

  *Init all the sensors.*
- void **RemoveSensor** (const std::string &_name)

  *Remove a sensor.*
- void **RemoveSensors** ()

  *Remove all sensors.*
- void **ResetLastUpdateTimes** ()

  *Reset last update times in all sensors.*
- void **RunThreads** ()

  *Run sensor updates in separate threads.*
- bool **SensorsInitialized** ()

  *True if SensorManager::initSensors queue is empty i.e.*
- void **Stop** ()

  *Stop the run thread.*
- void **Update** (bool _force=false)

  *Update all the sensors.*

## Additional Inherited Members

### 10.167.1 Detailed Description

Class to manage and update all sensors.

### 10.167.2 Member Function Documentation

#### 10.167.2.1 std::string gazebo::sensors::SensorManager::CreateSensor ( sdf::ElementPtr *_elem*, const std::string & *_worldName*, const std::string & *_parentName* )

Deprecated.

#### 10.167.2.2 std::string gazebo::sensors::SensorManager::CreateSensor ( sdf::ElementPtr *_elem*, const std::string & *_worldName*, const std::string & *_parentName*, uint32_t *_parentId* )

Add a sensor from an SDF element.

This function will also Load and Init the sensor.

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | The SDF element that describes the sensor |
| in | *_worldName* | Name of the world in which to create the sensor |
| in | *_parentName* | The name of the parent link which the sensor is attached to. |

**Returns**

The name of the sensor

#### 10.167.2.3 void gazebo::sensors::SensorManager::Fini ( )

Finalize all the sensors.

#### 10.167.2.4 SensorPtr gazebo::sensors::SensorManager::GetSensor ( const std::string & _name ) const

Get a sensor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of a sensor to find. |

**Returns**

A pointer to the sensor. NULL if not found.

#### 10.167.2.5 Sensor_V gazebo::sensors::SensorManager::GetSensors ( ) const

Get all the sensors.

**Returns**

Vector of all the sensors.

#### 10.167.2.6 void gazebo::sensors::SensorManager::GetSensorTypes ( std::vector< std::string > & _types ) const

Get all the sensor types.

**Parameters**

| | | |
|---|---|---|
| out | *All* | the sensor types. |

#### 10.167.2.7 void gazebo::sensors::SensorManager::Init ( )

Init all the sensors.

#### 10.167.2.8 void gazebo::sensors::SensorManager::RemoveSensor ( const std::string & _name )

Remove a sensor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the sensor to remove. |

#### 10.167.2.9 void gazebo::sensors::SensorManager::RemoveSensors ( )

Remove all sensors.

#### 10.167.2.10 void gazebo::sensors::SensorManager::ResetLastUpdateTimes ( )

Reset last update times in all sensors.

#### 10.167.2.11 void gazebo::sensors::SensorManager::RunThreads ( )

Run sensor updates in separate threads.

This will only run non-image based sensor updates.

#### 10.167.2.12 bool gazebo::sensors::SensorManager::SensorsInitialized ( )

True if SensorManager::initSensors queue is empty i.e.

all sensors managed by **SensorManager** (p. 850) have been initialized

#### 10.167.2.13 void gazebo::sensors::SensorManager::Stop ( )

Stop the run thread.

#### 10.167.2.14 void gazebo::sensors::SensorManager::Update ( bool *_force* = `false` )

Update all the sensors.

Checks to see if any sensor need to be initialized first, then updates all sensors once.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True force update, false if not |

The documentation for this class was generated from the following file:

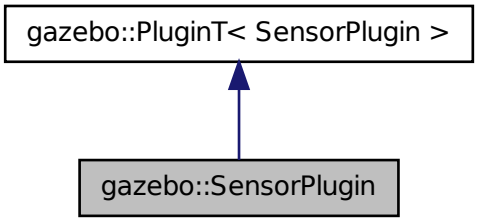
- **SensorManager.hh**

## 10.168 gazebo::SensorPlugin Class Reference

A plugin with access to physics::Sensor.

`#include <common/common.hh>`

Inheritance diagram for gazebo::SensorPlugin:




### Public Member Functions

- **SensorPlugin** ()

  *Constructor.*
- virtual ~**SensorPlugin** ()

  *Destructor.*
- virtual void **Init** ()

  *Override this method for custom plugin initialization behavior.*
- virtual void **Load** (**sensors::SensorPtr** _sensor, sdf::ElementPtr _sdf)=0

  *Load function.*
- virtual void **Reset** ()

  *Override this method for custom plugin reset behavior.*

### Additional Inherited Members

## 10.168.1 Detailed Description

A plugin with access to physics::Sensor.

See `reference`.

## 10.168.2 Constructor & Destructor Documentation

#### 10.168.2.1 gazebo::SensorPlugin::SensorPlugin ( ) [inline]

Constructor.

References gazebo::SENSOR_PLUGIN, and gazebo::PluginT< SensorPlugin >::type.

#### 10.168.2.2 virtual gazebo::SensorPlugin::~SensorPlugin ( ) [inline], [virtual]

Destructor.

### 10.168.3 Member Function Documentation

#### 10.168.3.1 virtual void gazebo::SensorPlugin::Init ( ) [inline], [virtual]

Override this method for custom plugin initialization behavior.

#### 10.168.3.2 virtual void gazebo::SensorPlugin::Load ( sensors::SensorPtr *_sensor,* sdf::ElementPtr *_sdf* ) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

**Parameters**

| | | |
|---|---|---|
| in | *_sensor* | Pointer the Sensor. |
| in | *_sdf* | Pointer the the SDF element of the plugin. |

#### 10.168.3.3 virtual void gazebo::SensorPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

- **Plugin.hh**

## 10.169 gazebo::Server Class Reference

`#include <Server.hh>`

### Public Member Functions

- **Server** ()
- virtual ~**Server** ()
- void **Fini** ()
- bool **GetInitialized** () const
- void **Init** ()
- bool **LoadFile** (const std::string &_filename="worlds/empty.world", const std::string &_physics="")

  *Load a world file and optionally override physics engine type.*
- bool **LoadString** (const std::string &_sdfString)
- bool **ParseArgs** (int argc, char **argv)
- bool **PreLoad** ()

  *Preload the server.*
- void **PrintUsage** ()
- void **Run** ()
- void **SetParams** (const **common::StrStr_M** &params)
- void **Stop** ()

### Public Attributes

- int **systemPluginsArgc**
- char ** **systemPluginsArgv**

## 10.169.1 Constructor & Destructor Documentation

#### 10.169.1.1 gazebo::Server::Server ( )

#### 10.169.1.2 virtual gazebo::Server::~Server ( ) [virtual]

## 10.169.2 Member Function Documentation

#### 10.169.2.1 void gazebo::Server::Fini ( )

#### 10.169.2.2 bool gazebo::Server::GetInitialized ( ) const

#### 10.169.2.3 void gazebo::Server::Init ( )

#### 10.169.2.4 bool gazebo::Server::LoadFile ( const std::string & *_filename* = "worlds/empty.world", const std::string & *_physics* = "" )

Load a world file and optionally override physics engine type.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | Name of the world file to load. |
| in | *_physics* | Physics engine type (ode\|bullet\|dart\|simbody). |

#### 10.169.2.5 bool gazebo::Server::LoadString ( const std::string & *_sdfString* )

#### 10.169.2.6 bool gazebo::Server::ParseArgs ( int *argc*, char ** *argv* )

#### 10.169.2.7 bool gazebo::Server::PreLoad ( )

Preload the server.

**Returns**

True if load was successful.

#### 10.169.2.8 void gazebo::Server::PrintUsage ( )

#### 10.169.2.9 void gazebo::Server::Run ( )

#### 10.169.2.10 void gazebo::Server::SetParams ( const common::StrStr_M & *params* )

#### 10.169.2.11 void gazebo::Server::Stop ( )

## 10.169.3 Member Data Documentation

#### 10.169.3.1 int gazebo::Server::systemPluginsArgc

#### 10.169.3.2 char** gazebo::Server::systemPluginsArgv

The documentation for this class was generated from the following file:

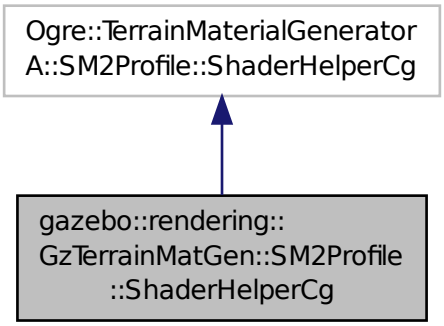
- **Server.hh**

## 10.170 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference

Keeping the CG shader for reference.

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg:




### Public Member Functions

- virtual Ogre::HighLevelGpuProgramPtr **generateFragmentProgram** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt)
- virtual Ogre::HighLevelGpuProgramPtr **generateVertexProgram** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt)

### Protected Member Functions

- virtual void **defaultVpParams** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr &_prog)
- virtual void **generateVertexProgramSource** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpDynamicShadows** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)

- virtual unsigned int **generateVpDynamicShadowsParams** (unsigned int _texCoordStart, const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpFooter** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpHeader** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)

### 10.170.1 Detailed Description

Keeping the CG shader for reference.

Utility class to help with generating shaders for Cg / HLSL.

### 10.170.2 Member Function Documentation

**10.170.2.1** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::defaultVpParams ( const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* const Ogre::HighLevelGpuProgramPtr & *_prog* ) [protected], [virtual]**

**10.170.2.2** **virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateFragmentProgram ( const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt* ) [virtual]**

**10.170.2.3** **virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgram ( const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt* ) [virtual]**

**10.170.2.4** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgramSource ( const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]**

**10.170.2.5** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows ( const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]**

**10.170.2.6** **virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadowsParams ( unsigned int *_texCoordStart,* const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]**

**10.170.2.7** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpFooter ( const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]**

**10.170.2.8** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpHeader ( const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]**

The documentation for this class was generated from the following file:

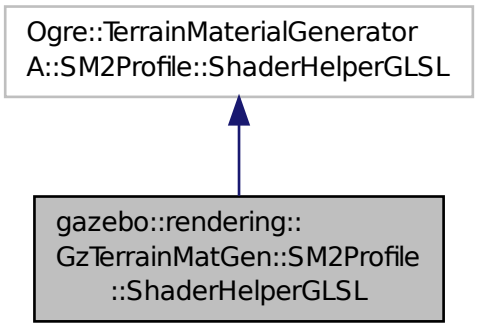
- **Heightmap.hh**

## 10.171 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference

Utility class to help with generating shaders for GLSL.

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL:




### Public Member Functions

- virtual Ogre::HighLevelGpuProgramPtr **generateFragmentProgram** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt)
- virtual Ogre::HighLevelGpuProgramPtr **generateVertexProgram** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt)
- virtual void **updateParams** (const **SM2Profile** ∗_prof, const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain, bool _compositeMap)

### Protected Member Functions

- virtual void **defaultVpParams** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr &_prog)
- void **generateFpDynamicShadows** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpDynamicShadowsHelpers** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpDynamicShadowsParams** (Ogre::uint ∗_texCoord, Ogre::uint ∗_sampler, const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpFooter** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpHeader** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType &_outStream)

- virtual void **generateFpLayer** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::uint _layer, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFragmentProgramSource** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVertexProgramSource** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpDynamicShadows** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual unsigned int **generateVpDynamicShadowsParams** (unsigned int _texCoordStart, const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpFooter** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpHeader** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **updateVpParams** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, const Ogre::GpuProgramParametersSharedPtr &_params)

### 10.171.1 Detailed Description

Utility class to help with generating shaders for GLSL.

### 10.171.2 Member Function Documentation

**10.171.2.1** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::defaultVpParams ( const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* const Ogre::HighLevelGpuProgramPtr & *_prog* )** [protected], [virtual]

**10.171.2.2** **void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadows ( const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* )** [protected]

**10.171.2.3** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsHelpers ( const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::StringUtil::StrStreamType & *_outStream* )** [protected], [virtual]

**10.171.2.4** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsParams ( Ogre::uint ∗ *_texCoord,* Ogre::uint ∗ *_sampler,* const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* )** [protected], [virtual]

**10.171.2.5** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpFooter ( const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::StringUtil::StrStreamType & *_outStream* )** [protected], [virtual]

**10.171.2.6** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpHeader ( const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::StringUtil::StrStreamType & *_outStream* )** [protected], [virtual]

**10.171.2.7** **virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpLayer ( const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::uint *_layer,* Ogre::StringUtil::StrStreamType & *_outStream* )** [protected], [virtual]

10.171.2.8 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL-::generateFragmentProgram ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt* ) [virtual]

10.171.2.9 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgramSource ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]

10.171.2.10 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL-::generateVertexProgram ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt* ) [virtual]

10.171.2.11 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgramSource ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]

10.171.2.12 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadows ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]

10.171.2.13 virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamic-ShadowsParams ( unsigned int *_texCoordStart,* const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]

10.171.2.14 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpFooter ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]

10.171.2.15 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpHeader ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream* ) [protected], [virtual]

10.171.2.16 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateParams ( const **SM2Profile** * *_prof,* const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain,* bool *_compositeMap* ) [virtual]

10.171.2.17 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateVpParams ( const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* const Ogre::GpuProgramParametersSharedPtr & *_params* ) [protected], [virtual]

The documentation for this class was generated from the following file:

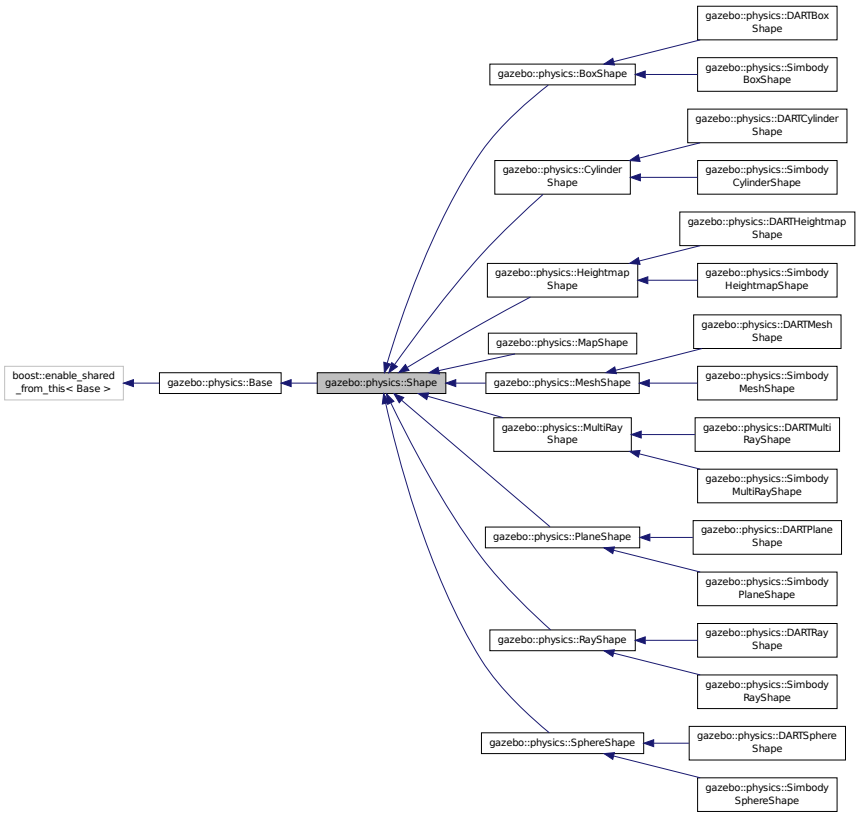
- **Heightmap.hh**

## 10.172 gazebo::physics::Shape Class Reference

**Base** (p. 159) class for all shapes.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Shape:




## Public Member Functions

- **Shape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**Shape** ()

  *Destructor.*
- virtual void **FillMsg** (msgs::Geometry &_msg)=0

  *Fill in the values for a geometry message.*
- virtual **math::Vector3 GetScale** () const

  *Get the scale of the shape.*
- virtual void **Init** ()=0

  *Initialize the shape.*
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0

  *Process a geometry message.*
- virtual void **SetScale** (const **math::Vector3** &_scale)=0

  *Set the scale of the shape.*

### Protected Attributes

- **CollisionPtr collisionParent**

  *This shape's collision parent.*
- **math::Vector3 scale**

  *This shape's scale;.*

### Additional Inherited Members

## 10.172.1 Detailed Description

**Base** (p. 159) class for all shapes.

## 10.172.2 Constructor & Destructor Documentation

#### 10.172.2.1 gazebo::physics::Shape::Shape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the shape. |

#### 10.172.2.2 virtual gazebo::physics::Shape::~Shape ( ) [virtual]

Destructor.

## 10.172.3 Member Function Documentation

#### 10.172.3.1 virtual void gazebo::physics::Shape::FillMsg ( msgs::Geometry & *_msg* ) [pure virtual]

Fill in the values for a geometry message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | The geometry message to fill. |

Implemented in **gazebo::physics::MultiRayShape** (p. 668), **gazebo::physics::RayShape** (p. 788), **gazebo::physics::HeightmapShape** (p. 468), **gazebo::physics::PlaneShape** (p. 730), **gazebo::physics::MeshShape** (p. 622), **gazebo::physics::CylinderShape** (p. 277), **gazebo::physics::MapShape** (p. 581), **gazebo::physics::SphereShape** (p. 987), and **gazebo::physics::BoxShape** (p. 178).

#### 10.172.3.2 virtual math::Vector3 gazebo::physics::Shape::GetScale ( ) const [virtual]

Get the scale of the shape.

**Returns**

Scale of the shape.

Reimplemented in **gazebo::physics::MapShape** (p. 582).

#### 10.172.3.3 virtual void gazebo::physics::Shape::Init ( ) [pure virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::Base** (p. 167).

Implemented in **gazebo::physics::RayShape** (p. 789), **gazebo::physics::MapShape** (p. 582), **gazebo::physics::HeightmapShape** (p. 469), **gazebo::physics::MeshShape** (p. 623), **gazebo::physics::MultiRayShape** (p. 671), **gazebo::physics::PlaneShape** (p. 731), **gazebo::physics::SphereShape** (p. 988), **gazebo::physics::BoxShape** (p. 178), **gazebo::physics::CylinderShape** (p. 278), **gazebo::physics::SimbodyHeightmapShape** (p. 879), **gazebo::physics::SimbodyMeshShape** (p. 911), **gazebo::physics::DARTHeightmapShape** (p. 294), and **gazebo::physics::DARTMeshShape** (p. 325).

#### 10.172.3.4 virtual void gazebo::physics::Shape::ProcessMsg ( const msgs::Geometry & *_msg* ) [pure virtual]

Process a geometry message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | The message to set values from. |

Implemented in **gazebo::physics::MultiRayShape** (p. 671), **gazebo::physics::RayShape** (p. 789), **gazebo::physics::HeightmapShape** (p. 470), **gazebo::physics::PlaneShape** (p. 731), **gazebo::physics::MeshShape** (p. 623), **gazebo::physics::CylinderShape** (p. 278), **gazebo::physics::MapShape** (p. 583), **gazebo::physics::SphereShape** (p. 988), and **gazebo::physics::BoxShape** (p. 178).

#### 10.172.3.5 virtual void gazebo::physics::Shape::SetScale ( const math::Vector3 & *_scale* ) [pure virtual]

Set the scale of the shape.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the shape to. |

Implemented in **gazebo::physics::RayShape** (p. 790), **gazebo::physics::MapShape** (p. 583), **gazebo::physics::PlaneShape** (p. 732), **gazebo::physics::MeshShape** (p. 623), **gazebo::physics::CylinderShape** (p. 278), **gazebo::physics::HeightmapShape** (p. 470), **gazebo::physics::SphereShape** (p. 988), **gazebo::physics::BoxShape** (p. 179), and **gazebo::physics::MultiRayShape** (p. 671).

### 10.172.4 Member Data Documentation

#### 10.172.4.1 CollisionPtr gazebo::physics::Shape::collisionParent [protected]

This shape's collision parent.

Referenced by gazebo::physics::DARTPlaneShape::CreatePlane(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo-

::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and gazebo::physics::-DARTBoxShape::SetSize().

#### 10.172.4.2 math::Vector3 gazebo::physics::Shape::scale [protected]

This shape's scale;.

The documentation for this class was generated from the following file:

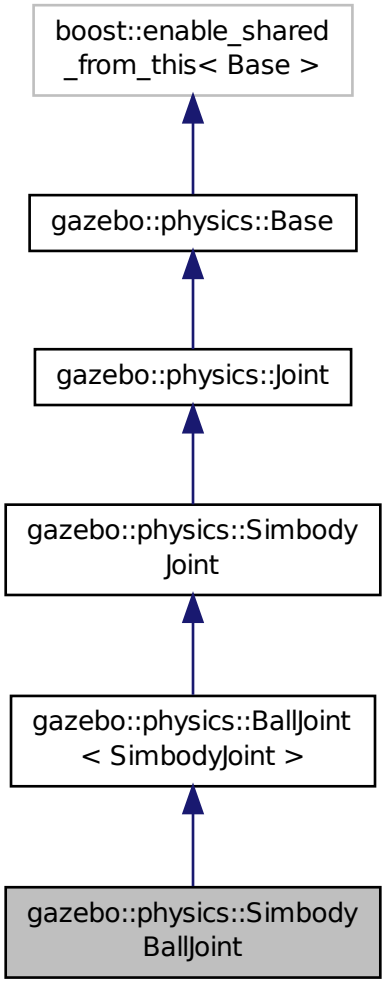
- **Shape.hh**

## 10.173 gazebo::physics::SimbodyBallJoint Class Reference

**SimbodyBallJoint** (p. 865) class models a ball joint in Simbody.

`#include <SimbodyBallJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyBallJoint:




## Public Member Functions

- **SimbodyBallJoint** (SimTK::MultibodySystem *_world, **BasePtr** _parent)

  *Simbody Ball **Joint** (p. 496) Constructor.*
- virtual ~**SimbodyBallJoint** ()

  *Destructor.*
- **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetAxis** (int) const

- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint** (p. 496) from a SDF sdf::Element.*
- virtual void **SetDamping** (int _index, double _damping)

  *Set joint damping, not yet implemented.*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*
- virtual void **SetMaxForce** (int _index, double _t)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _angle)

  *Set the velocity of an axis(index).*

## Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _torque)

  *Set the force applied to this **physics::Joint** (p. 496).*

## Additional Inherited Members

### 10.173.1 Detailed Description

**SimbodyBallJoint** (p. 865) class models a ball joint in Simbody.

### 10.173.2 Constructor & Destructor Documentation

#### 10.173.2.1 gazebo::physics::SimbodyBallJoint::SimbodyBallJoint ( SimTK::MultibodySystem * *_world,* BasePtr *_parent* )

Simbody Ball **Joint** (p. 496) Constructor.

#### 10.173.2.2 virtual gazebo::physics::SimbodyBallJoint::~SimbodyBallJoint ( ) [virtual]

Destructor.

### 10.173.3 Member Function Documentation

#### 10.173.3.1 math::Vector3 gazebo::physics::SimbodyBallJoint::GetAnchor ( int _index ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

Anchor value for the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 894).

#### 10.173.3.2 virtual math::Angle gazebo::physics::SimbodyBallJoint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.173.3.3 virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetAxis ( int ) const [inline], [virtual]

#### 10.173.3.4 virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | _index | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.173.3.5 virtual double gazebo::physics::SimbodyBallJoint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.173.3.6 virtual double gazebo::physics::SimbodyBallJoint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.173.3.7 virtual void gazebo::physics::SimbodyBallJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 510).

#### 10.173.3.8 virtual void gazebo::physics::SimbodyBallJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load **physics::Joint** (p. 496) from a SDF sdf::Element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 897).

#### 10.173.3.9 virtual void gazebo::physics::SimbodyBallJoint::SetDamping ( int *_index*, double *_damping* ) [virtual]

Set joint damping, not yet implemented.

**See Also**

**Joint::SetDamping** (p. 512)

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.173.3.10 virtual void gazebo::physics::SimbodyBallJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only. |

Implements **gazebo::physics::SimbodyJoint** (p. 899).

#### 10.173.3.11 virtual void gazebo::physics::SimbodyBallJoint::SetHighStop ( int *_index*, const **math::Angle** & *_angle* ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.173.3.12 virtual void gazebo::physics::SimbodyBallJoint::SetLowStop ( int *_index*, const **math::Angle** & *_angle* ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.173.3.13 virtual void gazebo::physics::SimbodyBallJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.173.3.14 virtual void gazebo::physics::SimbodyBallJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

The documentation for this class was generated from the following file:

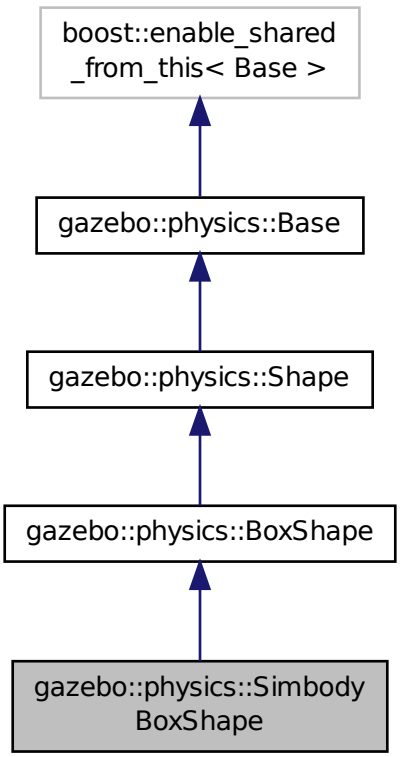
- **SimbodyBallJoint.hh**

## 10.174 gazebo::physics::SimbodyBoxShape Class Reference

Simbody box collision.

`#include <SimbodyBoxShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyBoxShape:



### Public Member Functions

- **SimbodyBoxShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**SimbodyBoxShape** ()

  *Destructor.*
- void **SetSize** (const **math::Vector3** &_size)

  *Set the size of the box.*

### Additional Inherited Members

#### 10.174.1 Detailed Description

Simbody box collision.

#### 10.174.2 Constructor & Destructor Documentation

##### 10.174.2.1 gazebo::physics::SimbodyBoxShape::SimbodyBoxShape ( CollisionPtr *_parent* ) [inline]

Constructor.

##### 10.174.2.2 virtual gazebo::physics::SimbodyBoxShape::~SimbodyBoxShape ( ) [inline], [virtual]

Destructor.

#### 10.174.3 Member Function Documentation

##### 10.174.3.1 void gazebo::physics::SimbodyBoxShape::SetSize ( const math::Vector3 & *_size* ) [inline], [virtual]

Set the size of the box.

**Parameters**

| | | |
|---|---|---|
| in | *_size* | Size of each side of the box. |

Reimplemented from **gazebo::physics::BoxShape** (p. 179).

References gazebo::physics::Shape::collisionParent, gazebo::math::equal(), gzerr, gzwarn, gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

The documentation for this class was generated from the following file:

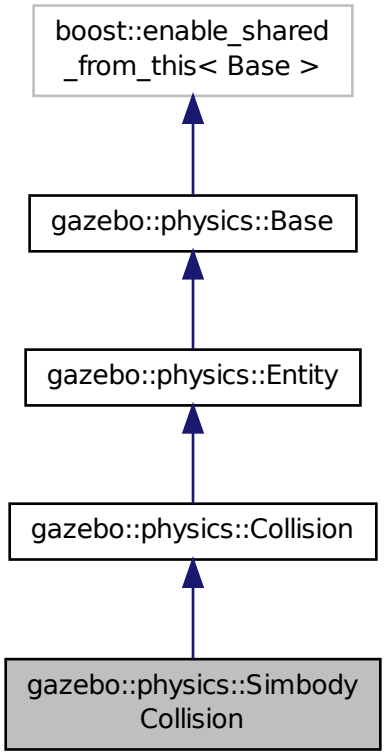
- **SimbodyBoxShape.hh**

## 10.175 gazebo::physics::SimbodyCollision Class Reference

Simbody collisions.

`#include <SimbodyCollision.hh>`

Inheritance diagram for gazebo::physics::SimbodyCollision:




## Public Member Functions

- **SimbodyCollision** (**LinkPtr** _parent)

  *Constructor.*
- virtual ∼**SimbodyCollision** ()

  *Destructor.*
- virtual **math::Box GetBoundingBox** () const

  *Get the bounding box for this collision.*
- SimTK::ContactGeometry ∗ **GetCollisionShape** () const

  *Get the simbody collision shape.*
- virtual void **Load** (sdf::ElementPtr _ptr)

  *Load the collision.*
- virtual void **OnPoseChange** ()

  *This function is called when the entity's (or one of its parents) pose of the parent has changed.*
- virtual void **SetCategoryBits** (unsigned int _bits)

  *Set the category bits, used during collision detection.*
- virtual void **SetCollideBits** (unsigned int _bits)

*Set the collide bits, used during collision detection.*

- void **SetCollisionShape** (SimTK::ContactGeometry *_shape)

  *Set the collision shape.*

## Additional Inherited Members

### 10.175.1 Detailed Description

Simbody collisions.

### 10.175.2 Constructor & Destructor Documentation

#### 10.175.2.1 gazebo::physics::SimbodyCollision::SimbodyCollision ( LinkPtr *_parent* )

Constructor.

#### 10.175.2.2 virtual gazebo::physics::SimbodyCollision::~SimbodyCollision ( ) [virtual]

Destructor.

### 10.175.3 Member Function Documentation

#### 10.175.3.1 virtual math::Box gazebo::physics::SimbodyCollision::GetBoundingBox ( ) const [virtual]

Get the bounding box for this collision.

**Returns**

The bounding box.

Implements **gazebo::physics::Collision** (p. 223).

#### 10.175.3.2 SimTK::ContactGeometry* gazebo::physics::SimbodyCollision::GetCollisionShape ( ) const

Get the simbody collision shape.

**Returns**

**SimTK** (p. 130) geometry used as the collision shape.

#### 10.175.3.3 virtual void gazebo::physics::SimbodyCollision::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF to load from. |

Reimplemented from **gazebo::physics::Collision** (p. 227).

#### 10.175.3.4 virtual void gazebo::physics::SimbodyCollision::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **gazebo::physics::Entity** (p. 386).

#### 10.175.3.5 virtual void gazebo::physics::SimbodyCollision::SetCategoryBits ( unsigned int *_bits* ) [virtual]

Set the category bits, used during collision detection.

**Parameters**

| | | |
|---|---|---|
| in | *_bits* | The bits to set. |

Implements **gazebo::physics::Collision** (p. 227).

#### 10.175.3.6 virtual void gazebo::physics::SimbodyCollision::SetCollideBits ( unsigned int *_bits* ) [virtual]

Set the collide bits, used during collision detection.

**Parameters**

| | | |
|---|---|---|
| in | *_bits* | The bits to set. |

Implements **gazebo::physics::Collision** (p. 227).

#### 10.175.3.7 void gazebo::physics::SimbodyCollision::SetCollisionShape ( SimTK::ContactGeometry * *_shape* )

Set the collision shape.

**Parameters**

| | | |
|---|---|---|
| in | *_shape* | **SimTK** (p. 130) geometry to use as the collision **SimTK** (p. 130) geometry to use as the collision shape. |

The documentation for this class was generated from the following file:

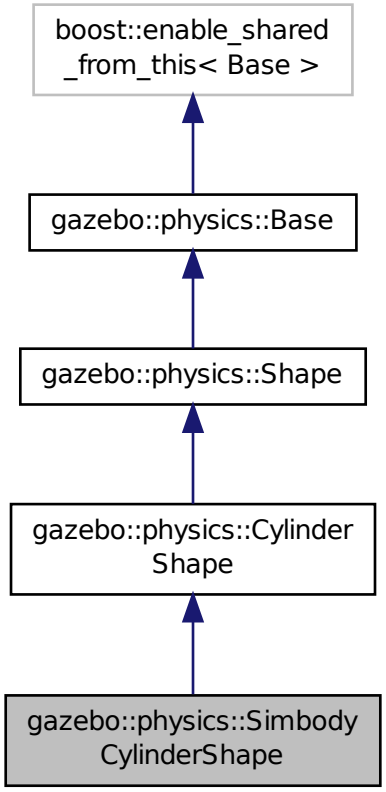
- **SimbodyCollision.hh**

## 10.176 gazebo::physics::SimbodyCylinderShape Class Reference

Cylinder collision.

`#include <SimbodyCylinderShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyCylinderShape:




## Public Member Functions

- **SimbodyCylinderShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**SimbodyCylinderShape** ()

  *Destructor.*
- void **SetSize** (double _radius, double _length)

  *Set the size of the cylinder.*

## Additional Inherited Members

### 10.176.1 Detailed Description

Cylinder collision.

### 10.176.2 Constructor & Destructor Documentation

#### 10.176.2.1 gazebo::physics::SimbodyCylinderShape::SimbodyCylinderShape ( CollisionPtr *_parent* ) [inline]

Constructor.

#### 10.176.2.2 virtual gazebo::physics::SimbodyCylinderShape::~SimbodyCylinderShape ( ) [inline], [virtual]

Destructor.

### 10.176.3 Member Function Documentation

#### 10.176.3.1 void gazebo::physics::SimbodyCylinderShape::SetSize ( double *_radius,* double *_length* ) [inline], [virtual]

Set the size of the cylinder.

**Parameters**

| | | |
|---|---|---|
| in | *_radius* | New radius. |
| in | *_lenght* | New length. |

Reimplemented from **gazebo::physics::CylinderShape** (p. 279).

References gazebo::physics::Shape::collisionParent, gazebo::math::equal(), gzerr, gzwarn, and gazebo::physics::CylinderShape::SetSize().

The documentation for this class was generated from the following file:

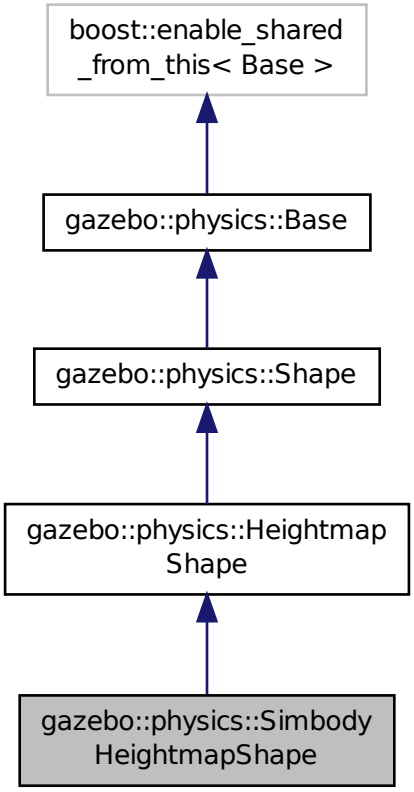
- **SimbodyCylinderShape.hh**

## 10.177 gazebo::physics::SimbodyHeightmapShape Class Reference

Height map collision.

`#include <SimbodyHeightmapShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyHeightmapShape:




## Public Member Functions

- **SimbodyHeightmapShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**SimbodyHeightmapShape** ()

  *Destructor.*
- virtual void **Init** ()

  *Initialize the heightmap.*

## Additional Inherited Members

### 10.177.1 Detailed Description

Height map collision.

### 10.177.2 Constructor & Destructor Documentation

#### 10.177.2.1 gazebo::physics::SimbodyHeightmapShape::SimbodyHeightmapShape ( CollisionPtr *_parent* )

Constructor.

#### 10.177.2.2 virtual gazebo::physics::SimbodyHeightmapShape::~SimbodyHeightmapShape ( ) [virtual]

Destructor.

### 10.177.3 Member Function Documentation

#### 10.177.3.1 virtual void gazebo::physics::SimbodyHeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 469).

The documentation for this class was generated from the following file:

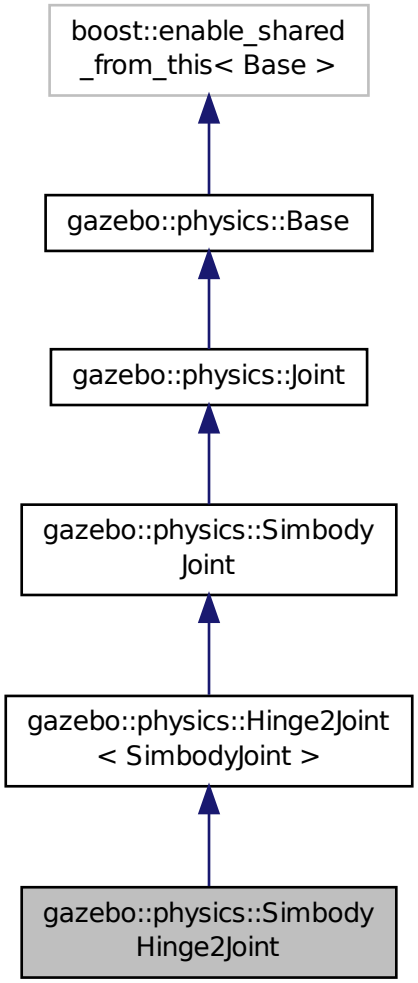
- **SimbodyHeightmapShape.hh**

## 10.178 gazebo::physics::SimbodyHinge2Joint Class Reference

A two axis hinge joint.

`#include <SimbodyHinge2Joint.hh>`

Inheritance diagram for gazebo::physics::SimbodyHinge2Joint:




## Public Member Functions

- **SimbodyHinge2Joint** (SimTK::MultibodySystem ∗**world**, **BasePtr** _parent)

  *Constructor.*
- virtual ∼**SimbodyHinge2Joint** ()

  *Destructor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual **math::Vector3 GetAxis** (int _index) const
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*

- virtual **math::Angle GetHighStop** (int _index)

  *Get the high stop of an axis(index).*
- virtual **math::Angle GetLowStop** (int _index)

  *Get the low stop of an axis(index).*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, double _damping)

  *Set joint damping, not yet implemented.*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*
- virtual void **SetMaxForce** (int _index, double _t)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _angle)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the joint.*
- virtual void **SetForceImpl** (int _index, double _torque)

  *Set the torque.*

### Additional Inherited Members

## 10.178.1 Detailed Description

A two axis hinge joint.

## 10.178.2 Constructor & Destructor Documentation

#### 10.178.2.1 gazebo::physics::SimbodyHinge2Joint::SimbodyHinge2Joint ( SimTK::MultibodySystem * *world,* BasePtr *_parent* )

Constructor.

#### 10.178.2.2 virtual gazebo::physics::SimbodyHinge2Joint::~SimbodyHinge2Joint ( ) [virtual]

Destructor.

### 10.178.3 Member Function Documentation

#### 10.178.3.1 virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetAnchor ( int *_index* ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Anchor value for the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 894).

#### 10.178.3.2 virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetAngleImpl ( int *_index* ) const [protected], [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.178.3.3 virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetAxis ( int *_index* ) const [virtual]

#### 10.178.3.4 virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetGlobalAxis ( int *_index* ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.178.3.5 virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetHighStop ( int *_index* ) [virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 506).

#### 10.178.3.6 virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetLowStop ( int *_index* ) [virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.178.3.7 virtual double gazebo::physics::SimbodyHinge2Joint::GetMaxForce ( int *_index* ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.178.3.8 virtual double gazebo::physics::SimbodyHinge2Joint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.178.3.9 virtual void gazebo::physics::SimbodyHinge2Joint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 510).

#### 10.178.3.10 virtual void gazebo::physics::SimbodyHinge2Joint::Load ( sdf::ElementPtr *_sdf* ) [protected], [virtual]

Load the joint.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 472).

#### 10.178.3.11 virtual void gazebo::physics::SimbodyHinge2Joint::SetAxis ( int *_index*, const math::Vector3 & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.178.3.12 virtual void gazebo::physics::SimbodyHinge2Joint::SetDamping ( int *_index*, double *_damping* ) [virtual]

Set joint damping, not yet implemented.

**See Also**

Hinge2Joint::SetDamping

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.178.3.13 virtual void gazebo::physics::SimbodyHinge2Joint::SetForceImpl ( int *_index*, double *_torque* ) [protected], [virtual]

Set the torque.

Implements **gazebo::physics::SimbodyJoint** (p. 899).

#### 10.178.3.14 virtual void gazebo::physics::SimbodyHinge2Joint::SetHighStop ( int _index, const math::Angle & _angle ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.178.3.15 virtual void gazebo::physics::SimbodyHinge2Joint::SetLowStop ( int _index, const math::Angle & _angle ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.178.3.16 virtual void gazebo::physics::SimbodyHinge2Joint::SetMaxForce ( int _index, double _force ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.178.3.17 virtual void gazebo::physics::SimbodyHinge2Joint::SetVelocity ( int _index, double _vel ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

The documentation for this class was generated from the following file:

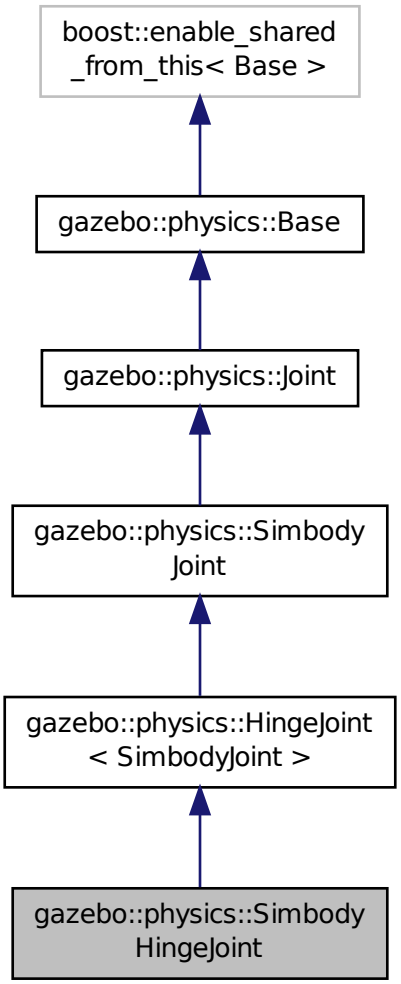
- **SimbodyHinge2Joint.hh**

## 10.179 gazebo::physics::SimbodyHingeJoint Class Reference

A single axis hinge joint.

`#include <SimbodyHingeJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyHingeJoint:




### Public Member Functions

- **SimbodyHingeJoint** (SimTK::MultibodySystem ∗**world**, **BasePtr** _parent)

  *Constructor.*
- virtual ∼**SimbodyHingeJoint** ()

  *Destructor.*

- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*
- virtual **math::Angle GetHighStop** (int _index)

  *Get the high stop of an axis(index).*
- virtual **math::Angle GetLowStop** (int _index)

  *Get the low stop of an axis(index).*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **RestoreSimbodyState** (SimTK::State &_state)

  *restore simbody state for spawning*
- virtual void **SaveSimbodyState** (const SimTK::State &_state)

  *save simbody state for spawning*
- void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, double _damping)

  *Set the joint damping.*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*
- virtual void **SetMaxForce** (int _index, double _t)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _rate)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load joint.*
- virtual void **SetForceImpl** (int _index, double _torque)

  *Set the force applied to this **physics::Joint** (p. 496).*

### Additional Inherited Members

## 10.179.1 Detailed Description

A single axis hinge joint.

## 10.179.2 Constructor & Destructor Documentation

#### 10.179.2.1 gazebo::physics::SimbodyHingeJoint::SimbodyHingeJoint ( SimTK::MultibodySystem * *world,* BasePtr *_parent* )

Constructor.

#### 10.179.2.2 virtual gazebo::physics::SimbodyHingeJoint::~SimbodyHingeJoint ( ) [virtual]

Destructor.

### 10.179.3 Member Function Documentation

#### 10.179.3.1 virtual math::Angle gazebo::physics::SimbodyHingeJoint::GetAngleImpl ( int _index ) const [protected], [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.179.3.2 virtual math::Vector3 gazebo::physics::SimbodyHingeJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.179.3.3 virtual math::Angle gazebo::physics::SimbodyHingeJoint::GetHighStop ( int _index ) [virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 506).

#### 10.179.3.4 virtual math::Angle gazebo::physics::SimbodyHingeJoint::GetLowStop ( int _index ) [virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.179.3.5 virtual double gazebo::physics::SimbodyHingeJoint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.179.3.6 virtual double gazebo::physics::SimbodyHingeJoint::GetVelocity ( int _index ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.179.3.7 virtual void gazebo::physics::SimbodyHingeJoint::Load ( sdf::ElementPtr _sdf ) [protected], [virtual]

Load joint.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to SDF element |

Reimplemented from **gazebo::physics::HingeJoint**< **SimbodyJoint** > (p. 474).

#### 10.179.3.8 virtual void gazebo::physics::SimbodyHingeJoint::RestoreSimbodyState ( SimTK::State & *_state* ) [virtual]

restore simbody state for spawning

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 897).

#### 10.179.3.9 virtual void gazebo::physics::SimbodyHingeJoint::SaveSimbodyState ( const SimTK::State & *_state* ) [virtual]

save simbody state for spawning

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 897).

#### 10.179.3.10 void gazebo::physics::SimbodyHingeJoint::SetAxis ( int *_index,* const **math::Vector3** & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.179.3.11 virtual void gazebo::physics::SimbodyHingeJoint::SetDamping ( int *_index,* double *_damping* ) [virtual]

Set the joint damping.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set, currently ignored, to be implemented. |
| in | *_damping* | Damping value for the axis. |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.179.3.12 virtual void gazebo::physics::SimbodyHingeJoint::SetForceImpl ( int *_index,* double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only. |

Implements **gazebo::physics::SimbodyJoint** (p. 899).

#### 10.179.3.13 virtual void gazebo::physics::SimbodyHingeJoint::SetHighStop ( int *_index*, const math::Angle & *_angle* ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.179.3.14 virtual void gazebo::physics::SimbodyHingeJoint::SetLowStop ( int *_index*, const math::Angle & *_angle* ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.179.3.15 virtual void gazebo::physics::SimbodyHingeJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.179.3.16 virtual void gazebo::physics::SimbodyHingeJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

The documentation for this class was generated from the following file:

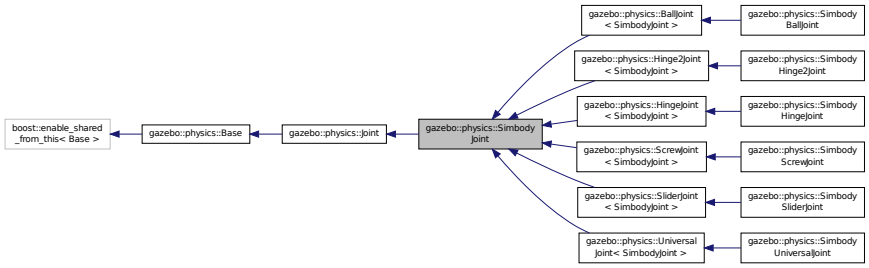
- **SimbodyHingeJoint.hh**

## 10.180 gazebo::physics::SimbodyJoint Class Reference

**Base** (p. 159) class for all joints.

`#include <SimbodyJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyJoint:




### Public Member Functions

- **SimbodyJoint** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**SimbodyJoint** ()

  *Destructor.*
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const

  *Determines of the two bodies are connected by a joint.*
- virtual void **CacheForceTorque** ()

  *Cache **Joint** (p. 496) Force Torque Values if necessary for physics engine.*
- virtual void **Detach** ()

  *Detach this joint from all links.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index)

  *Get a non-generic parameter for the joint.*
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)

  *get internal force and torque values at a joint.*
- virtual **LinkPtr GetJointLink** (int _index) const

  *Get the link to which the joint is attached according the _index.*
- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const

  *Get the forces applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const

  *Get the torque applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint** (p. 496) from a SDF sdf::Element.*
- virtual void **Reset** ()

  *Reset the joint.*
- virtual void **RestoreSimbodyState** (SimTK::State &_state)

- virtual void **SaveSimbodyState** (const SimTK::State &_state)
- virtual void **SetAnchor** (int _index, const **gazebo::math::Vector3** &_anchor)

  *Set the anchor point.*
- virtual void **SetAttribute** (**Attribute**, int _index, double _value)

  *Set a parameter for the joint.*
- virtual void **SetAttribute** (const std::string &_key, int _index, const boost::any &_value)

  *Set a non-generic parameter for the joint.*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, const double _damping)

  *Set the joint damping.*
- virtual void **SetForce** (int _index, double _force)

  *Set the force applied to this **physics::Joint*** (p. *496*).

## Public Attributes

- SimTK::Constraint **constraint**

  *: isValid() if we used a constraint to model this joint.*
- SimTK::Force::MobilityLinearDamper **damper**

  *: for enforcing joint damping forces.*
- SimTK::Transform **defxAB**

  *default mobilizer pose*
- bool **isReversed**

  *: if mobilizer, did it reverse parent&child? Set when we build the Simbody model.*
- SimTK::Force::MobilityLinearStop **limitForce**

  *: for enforcing joint stops Set when we build the Simbody model.*
- SimTK::MobilizedBody **mobod**

  *Use isValid() if we used a mobilizer Set when we build the Simbody model.*
- bool **mustBreakLoopHere**

  *Force Simbody to break a loop by using a weld constraint.*
- bool **physicsInitialized**
- SimTK::Transform **xCB**

  *child body frame to mobilizer frame*
- SimTK::Transform **xPA**

  *Normally A=F, B=M.*

## Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _force)=0

  *Set the force applied to this **physics::Joint*** (p. *496*).

## Protected Attributes

- **SimbodyPhysicsPtr simbodyPhysics**

  *keep a pointer to the simbody physics engine for convenience*
- SimTK::MultibodySystem * **world**

  *Simbody Multibody System.*

### Additional Inherited Members

## 10.180.1 Detailed Description

**Base** (p. 159) class for all joints.

## 10.180.2 Constructor & Destructor Documentation

### 10.180.2.1 gazebo::physics::SimbodyJoint::SimbodyJoint ( BasePtr _parent )

Constructor.

### 10.180.2.2 virtual gazebo::physics::SimbodyJoint::~SimbodyJoint ( ) [virtual]

Destructor.

## 10.180.3 Member Function Documentation

### 10.180.3.1 virtual bool gazebo::physics::SimbodyJoint::AreConnected ( LinkPtr _one, LinkPtr _two ) const [virtual]

Determines of the two bodies are connected by a joint.

**Parameters**

| | | |
|---|---|---|
| in | *_one* | First link. |
| in | *_two* | Second link. |

**Returns**

True if the two links are connected by a joint.

Implements **gazebo::physics::Joint** (p. 501).

### 10.180.3.2 virtual void gazebo::physics::SimbodyJoint::CacheForceTorque ( ) [virtual]

Cache **Joint** (p. 496) Force Torque Values if necessary for physics engine.

Reimplemented from **gazebo::physics::Joint** (p. 501).

### 10.180.3.3 virtual void gazebo::physics::SimbodyJoint::Detach ( ) [virtual]

Detach this joint from all links.

Reimplemented from **gazebo::physics::Joint** (p. 502).

### 10.180.3.4 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetAnchor ( int _index ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 502).

Reimplemented in **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 974), **gazebo::physics::SimbodyHinge2Joint** (p. 882), **gazebo::physics::SimbodyUniversalJoint** (p. 947), and **gazebo::physics::SimbodyBallJoint** (p. 868).

#### 10.180.3.5 virtual double gazebo::physics::SimbodyJoint::GetAttribute ( const std::string & *_key,* unsigned int *_index* ) [virtual]

Get a non-generic parameter for the joint.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | String key. |
| in | *_index* | Index of the axis. |
| in | *_value* | Value of the attribute. |

Implements **gazebo::physics::Joint** (p. 504).

#### 10.180.3.6 virtual double gazebo::physics::SimbodyJoint::GetForce ( unsigned int *_index* ) [virtual]

**Todo** : not yet implemented. Get external forces applied at this **Joint** (p. 496). Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The force applied to an axis.

Reimplemented from **gazebo::physics::Joint** (p. 505).

#### 10.180.3.7 virtual JointWrench gazebo::physics::SimbodyJoint::GetForceTorque ( unsigned int *_index* ) [virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 529) data structure. Where **JointWrench.body1Force** (p. 530) contains the force applied by the parent **Link** (p. 542) on the **Joint** (p. 496) specified in the parent **Link** (p. 542) frame, and **JointWrench.body2Force** (p. 531) contains the force applied by the child **Link** (p. 542) on the **Joint** (p. 496) specified in the child **Link** (p. 542) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 496) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 542) frames.

Note that for ODE you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Not used right now |

**Returns**

The force and torque at the joint, see above for details on conventions.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.180.3.8 virtual LinkPtr gazebo::physics::SimbodyJoint::GetJointLink ( int *_index* ) const [virtual]

Get the link to which the joint is attached according the _index.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the link to retreive. |

**Returns**

Pointer to the request link. NULL if the index was invalid.

Implements **gazebo::physics::Joint** (p. 507).

#### 10.180.3.9 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetLinkForce ( unsigned int *_index* ) const [virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of the link(0 or 1). |

**Returns**

Force applied to the link.

Implements **gazebo::physics::Joint** (p. 507).

#### 10.180.3.10 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetLinkTorque ( unsigned int *_index* ) const [virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 542) due to the existence of this **Joint** (p. 496).

Note that the unit of torque should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of the link(0 or 1) |

**Returns**

Torque applied to the link.

Implements **gazebo::physics::Joint** (p. 507).

#### 10.180.3.11 virtual void gazebo::physics::SimbodyJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load **physics::Joint** (p. 496) from a SDF sdf::Element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::Joint** (p. 510).

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 940), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 472), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 158), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1065), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 474), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 975), **gazebo::physics::SimbodyHingeJoint** (p. 889), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::SimbodyScrewJoint** (p. 934), and **gazebo::physics::SimbodyBallJoint** (p. 869).

#### 10.180.3.12 virtual void gazebo::physics::SimbodyJoint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 511).

#### 10.180.3.13 virtual void gazebo::physics::SimbodyJoint::RestoreSimbodyState ( SimTK::State & *_state* ) [virtual]

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 890).

#### 10.180.3.14 virtual void gazebo::physics::SimbodyJoint::SaveSimbodyState ( const SimTK::State & *_state* ) [virtual]

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 890).

#### 10.180.3.15 virtual void gazebo::physics::SimbodyJoint::SetAnchor ( int *_index*, const gazebo::math::Vector3 & *_anchor* ) [virtual]

Set the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Indx of the axis. |
| in | *_anchor* | Anchor value. |

Implements **gazebo::physics::Joint** (p. 511).

Reimplemented in **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834), and **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 975).

#### 10.180.3.16 virtual void gazebo::physics::SimbodyJoint::SetAttribute ( Attribute , int _index, double _value ) [virtual]

Set a parameter for the joint.

#### 10.180.3.17 virtual void gazebo::physics::SimbodyJoint::SetAttribute ( const std::string & _key, int _index, const boost::any & _value ) [virtual]

Set a non-generic parameter for the joint.

replaces **SetAttribute(Attribute, int, double)** (p. 898)

**Parameters**

| | | |
|---|---|---|
| in | *_key* | String key. |
| in | *_index* | Index of the axis. |
| in | *_value* | Value of the attribute. |

Implements **gazebo::physics::Joint** (p. 511).

#### 10.180.3.18 virtual void gazebo::physics::SimbodyJoint::SetAxis ( int _index, const math::Vector3 & _axis ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Implements **gazebo::physics::Joint** (p. 512).

Reimplemented in **gazebo::physics::BallJoint< SimbodyJoint >** (p. 159), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::SimbodyScrewJoint** (p. 934), **gazebo::physics::SimbodyHingeJoint** (p. 890), and **gazebo::physics::SimbodySliderJoint** (p. 941).

#### 10.180.3.19 virtual void gazebo::physics::SimbodyJoint::SetDamping ( int _index, const double _damping ) [virtual]

Set the joint damping.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set, currently ignored, to be implemented. |
| in | *_damping* | Damping value for the axis. |

Implements **gazebo::physics::Joint** (p. 512).

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 941), **gazebo::physics::SimbodyUniversalJoint** (p. 949), **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyHingeJoint** (p. 890), **gazebo::physics::SimbodyScrewJoint** (p. 934), and **gazebo::physics::SimbodyBallJoint** (p. 869).

#### 10.180.3.20 virtual void gazebo::physics::SimbodyJoint::SetForce ( int _index, double _effort ) [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to

SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)). Forces are truncated by effortLimit before applied.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_effort* | Force value. |

Implements **gazebo::physics::Joint** (p. 512).

#### 10.180.3.21 virtual void gazebo::physics::SimbodyJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [pure virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only. |

Implemented in **gazebo::physics::SimbodyHinge2Joint** (p. 884), **gazebo::physics::SimbodyUniversalJoint** (p. 950), **gazebo::physics::SimbodyScrewJoint** (p. 935), **gazebo::physics::SimbodyHingeJoint** (p. 890), **gazebo::physics::SimbodySliderJoint** (p. 941), and **gazebo::physics::SimbodyBallJoint** (p. 870).

### 10.180.4 Member Data Documentation

#### 10.180.4.1 SimTK::Constraint gazebo::physics::SimbodyJoint::constraint

: isValid() if we used a constraint to model this joint.

Set when we build the Simbody model. How this joint was modeled in the Simbody System. We used either a mobilizer or a constraint, but not both. The type of either one is the same as the joint type above.

#### 10.180.4.2 SimTK::Force::MobilityLinearDamper gazebo::physics::SimbodyJoint::damper

: for enforcing joint damping forces.

Set when we build the Simbody model. : Make these arrays for multi-axis joints. : Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints.

#### 10.180.4.3 SimTK::Transform gazebo::physics::SimbodyJoint::defxAB

default mobilizer pose

#### 10.180.4.4 bool gazebo::physics::SimbodyJoint::isReversed

: if mobilizer, did it reverse parent&child? Set when we build the Simbody model.

#### 10.180.4.5 SimTK::Force::MobilityLinearStop gazebo::physics::SimbodyJoint::limitForce

: for enforcing joint stops Set when we build the Simbody model.

: Make these arrays for multi-axis joints. : Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints.

#### 10.180.4.6 SimTK::MobilizedBody gazebo::physics::SimbodyJoint::mobod

Use isValid() if we used a mobilizer Set when we build the Simbody model.

How this joint was modeled in the Simbody System. We used either a mobilizer or a constraint, but not both. The type of either one is the same as the joint type above.

#### 10.180.4.7 bool gazebo::physics::SimbodyJoint::mustBreakLoopHere

Force Simbody to break a loop by using a weld constraint.

This flag is needed by SimbodyPhysics::MultibodyGraphMaker, so kept public.

#### 10.180.4.8 bool gazebo::physics::SimbodyJoint::physicsInitialized

#### 10.180.4.9 SimbodyPhysicsPtr gazebo::physics::SimbodyJoint::simbodyPhysics [protected]

keep a pointer to the simbody physics engine for convenience

#### 10.180.4.10 SimTK::MultibodySystem* gazebo::physics::SimbodyJoint::world [protected]

Simbody Multibody System.

#### 10.180.4.11 SimTK::Transform gazebo::physics::SimbodyJoint::xCB

child body frame to mobilizer frame

#### 10.180.4.12 SimTK::Transform gazebo::physics::SimbodyJoint::xPA

Normally A=F, B=M.

But if reversed, then B=F, A=M. parent body frame to mobilizer frame

The documentation for this class was generated from the following file:

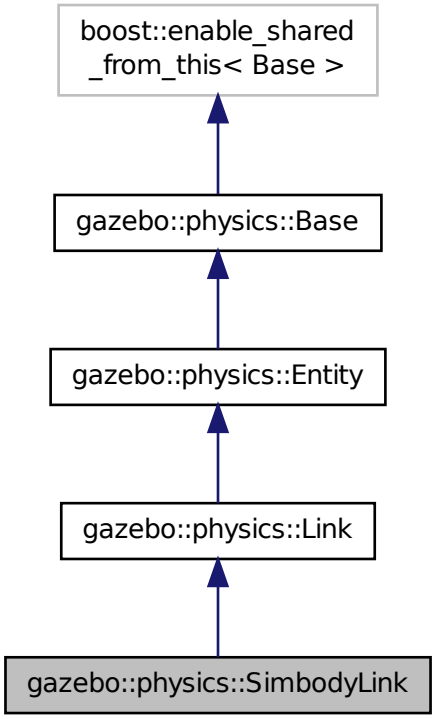
- **SimbodyJoint.hh**

## 10.181 gazebo::physics::SimbodyLink Class Reference

Simbody **Link** (p. 542) class.

`#include <SimbodyLink.hh>`

Inheritance diagram for gazebo::physics::SimbodyLink:




## Public Member Functions

- **SimbodyLink** (**EntityPtr** _parent)

  *Constructor.*
- virtual ~**SimbodyLink** ()

  *Destructor.*
- virtual void **AddForce** (const **math::Vector3** &_force)

  *Add a force to the body.*
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

  *Add a force to the body at position expressed to the body's own frame of reference.*
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

  *Add a force to the body using a global position.*
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

  *Add a force to the body, components are relative to the body's own frame of reference.*
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

  *Add a torque to the body, components are relative to the body's own frame of reference.*
- virtual void **AddTorque** (const **math::Vector3** &_torque)

  *Add a torque to the body.*

- virtual void **Fini** ()

  *Finalize the body.*
- SimTK::MassProperties **GetEffectiveMassProps** (int _numFragments) const
- virtual bool **GetEnabled** () const

  *Get whether this body is enabled in the physics engine.*
- virtual bool **GetGravityMode** () const

  *Get the gravity mode.*
- SimTK::MassProperties **GetMassProperties** () const

  *Convert Gazebo Inertia to Simbody MassProperties Where Simbody MassProperties contains mass, center of mass location, and unit inertia about body origin.*
- virtual **math::Vector3 GetWorldAngularVel** () const

  *Get the angular velocity of the entity in the world frame.*
- virtual **math::Vector3 GetWorldCoGLinearVel** () const

  *Get the linear velocity at the body's center of gravity in the world frame.*
- virtual **math::Vector3 GetWorldForce** () const

  *Get the force applied to the body in the world frame.*
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_vector3) const

  *Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.*
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset, const **math::Quaternion** &_q) const

  *Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.*
- virtual **math::Vector3 GetWorldTorque** () const

  *Get the torque applied to the body in the world frame.*
- virtual void **Init** ()

  *Initialize the body.*
- virtual void **Load** (sdf::ElementPtr _ptr)

  *Load the body based on an SDF element.*
- virtual void **OnPoseChange** ()

  *This function is called when the entity's (or one of its parents) pose of the parent has changed.*
- virtual void **RestoreSimbodyState** (SimTK::State &_state)
- virtual void **SaveSimbodyState** (const SimTK::State &_state)
- virtual void **SetAngularDamping** (double _damping)

  *Set the angular damping factor.*
- virtual void **SetAngularVel** (const **math::Vector3** &_vel)

  *Set the angular velocity of the body.*
- virtual void **SetAutoDisable** (bool _disable)

  *Allow the link to auto disable.*
- void **SetDirtyPose** (const **math::Pose** &_pose)
- virtual void **SetEnabled** (bool enable) const

  *Set whether this body is enabled.*
- virtual void **SetForce** (const **math::Vector3** &_force)

  *Set the force applied to the body.*
- virtual void **SetGravityMode** (bool _mode)

  *Set whether gravity affects this body.*
- virtual void **SetLinearDamping** (double _damping)

  *Set the linear damping factor.*
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)

  *Set the linear velocity of the body.*

- virtual void **SetLinkStatic** (bool _static)

  *If the inboard body of this link is ground, simply lock the inboard joint to freeze it to ground.*
- virtual void **SetSelfCollide** (bool _collide)

  *Set whether this body will collide with others in the model.*
- virtual void **SetTorque** (const **math::Vector3** &_force)

  *Set the torque applied to the body.*

### Public Attributes

- SimTK::MobilizedBody **masterMobod**
- bool **mustBeBaseLink**

  *: Force this link to be a base body, where its inboard body is the world with 6DOF.*
- bool **physicsInitialized**
- std::vector< SimTK::MobilizedBody > **slaveMobods**
- std::vector
  < SimTK::Constraint::Weld > **slaveWelds**

### Additional Inherited Members

## 10.181.1 Detailed Description

Simbody **Link** (p. 542) class.

## 10.181.2 Constructor & Destructor Documentation

#### 10.181.2.1 gazebo::physics::SimbodyLink::SimbodyLink ( EntityPtr _parent )

Constructor.

#### 10.181.2.2 virtual gazebo::physics::SimbodyLink::~SimbodyLink ( ) [virtual]

Destructor.

## 10.181.3 Member Function Documentation

#### 10.181.3.1 virtual void gazebo::physics::SimbodyLink::AddForce ( const math::Vector3 & _force ) [virtual]

Add a force to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |

Implements **gazebo::physics::Link** (p. 547).

#### 10.181.3.2 virtual void gazebo::physics::SimbodyLink::AddForceAtRelativePosition ( const math::Vector3 & *_force,* const math::Vector3 & *_relPos* ) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |
| in | *_relPos* | Position on the link to add the force. |

Implements **gazebo::physics::Link** (p. 547).

#### 10.181.3.3 virtual void gazebo::physics::SimbodyLink::AddForceAtWorldPosition ( const math::Vector3 & *_force,* const math::Vector3 & *_pos* ) [virtual]

Add a force to the body using a global position.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |
| in | *_pos* | Position in global coord frame to add the force. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.181.3.4 virtual void gazebo::physics::SimbodyLink::AddRelativeForce ( const math::Vector3 & *_force* ) [virtual]

Add a force to the body, components are relative to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force to add. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.181.3.5 virtual void gazebo::physics::SimbodyLink::AddRelativeTorque ( const math::Vector3 & *_torque* ) [virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value to add. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.181.3.6 virtual void gazebo::physics::SimbodyLink::AddTorque ( const math::Vector3 & *_torque* ) [virtual]

Add a torque to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value to add to the link. |

Implements **gazebo::physics::Link** (p. 548).

#### 10.181.3.7 virtual void gazebo::physics::SimbodyLink::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Link** (p. 550).

#### 10.181.3.8 SimTK::MassProperties gazebo::physics::SimbodyLink::GetEffectiveMassProps ( int *_numFragments* ) const

#### 10.181.3.9 virtual bool gazebo::physics::SimbodyLink::GetEnabled ( ) const [virtual]

Get whether this body is enabled in the physics engine.

**Returns**

True if the link is enabled.

Implements **gazebo::physics::Link** (p. 551).

#### 10.181.3.10 virtual bool gazebo::physics::SimbodyLink::GetGravityMode ( ) const [virtual]

Get the gravity mode.

**Returns**

True if gravity is enabled.

Implements **gazebo::physics::Link** (p. 551).

#### 10.181.3.11 SimTK::MassProperties gazebo::physics::SimbodyLink::GetMassProperties ( ) const

Convert Gazebo Inertia to Simbody MassProperties Where Simbody MassProperties contains mass, center of mass location, and unit inertia about body origin.

#### 10.181.3.12 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

**Returns**

A **math::Vector3** (p. 1091) for the velocity.

Reimplemented from **gazebo::physics::Entity** (p. 385).

#### 10.181.3.13 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldCoGLinearVel ( ) const [virtual]

Get the linear velocity at the body's center of gravity in the world frame.

**Returns**

Linear velocity at the body's center of gravity in the world frame.

Implements **gazebo::physics::Link** (p. 555).

#### 10.181.3.14 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldForce ( ) const [virtual]

Get the force applied to the body in the world frame.

**Returns**

Force applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 555).

#### 10.181.3.15 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldLinearVel ( const math::Vector3 & _offset ) const [virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.

If no offset is given, the velocity at the origin of the **Link** (p. 542) frame will be returned.

**Parameters**

| | | |
|---|---|---|
| in | *_offset* | Offset of the point from the origin of the **Link** (p. 542) frame, expressed in the body-fixed frame. |

**Returns**

Linear velocity of the point on the body

Implements **gazebo::physics::Link** (p. 555).

#### 10.181.3.16 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldLinearVel ( const math::Vector3 & _offset, const math::Quaternion & _q ) const [virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

**Parameters**

| | | |
|---|---|---|
| in | *_offset* | Offset from the origin of the link frame expressed in a frame defined by _q. |
| in | *_q* | Describes the rotation of a reference frame relative to the world reference frame. |

**Returns**

Linear velocity of the point on the body in the world frame.

Implements **gazebo::physics::Link** (p. 556).

#### 10.181.3.17 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldTorque ( ) const [virtual]

Get the torque applied to the body in the world frame.

**Returns**

Torque applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 556).

#### 10.181.3.18 virtual void gazebo::physics::SimbodyLink::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Link** (p. 556).

#### 10.181.3.19 virtual void gazebo::physics::SimbodyLink::Load ( sdf::ElementPtr _sdf ) [virtual]

Load the body based on an SDF element.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters. |

Reimplemented from **gazebo::physics::Link** (p. 556).

#### 10.181.3.20 virtual void gazebo::physics::SimbodyLink::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Reimplemented from **gazebo::physics::Link** (p. 557).

#### 10.181.3.21 virtual void gazebo::physics::SimbodyLink::RestoreSimbodyState ( SimTK::State & _state ) [virtual]

#### 10.181.3.22 virtual void gazebo::physics::SimbodyLink::SaveSimbodyState ( const SimTK::State & _state ) [virtual]

#### 10.181.3.23 virtual void gazebo::physics::SimbodyLink::SetAngularDamping ( double _damping ) [virtual]

Set the angular damping factor.

**Parameters**

| | | |
|---|---|---|
| in | *_damping* | Angular damping factor. |

Implements **gazebo::physics::Link** (p. 558).

#### 10.181.3.24 virtual void gazebo::physics::SimbodyLink::SetAngularVel ( const math::Vector3 & _vel ) [virtual]

Set the angular velocity of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Angular velocity. |

Implements **gazebo::physics::Link** (p. 558).

#### 10.181.3.25 virtual void gazebo::physics::SimbodyLink::SetAutoDisable ( bool _disable ) [virtual]

Allow the link to auto disable.

**Parameters**

| | | |
|---|---|---|
| in | *_disable* | If true, the link is allowed to auto disable. |

Implements **gazebo::physics::Link** (p. 558).

#### 10.181.3.26 void gazebo::physics::SimbodyLink::SetDirtyPose ( const math::Pose & *_pose* )

#### 10.181.3.27 virtual void gazebo::physics::SimbodyLink::SetEnabled ( bool *_enable* ) const [virtual]

Set whether this body is enabled.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | True to enable the link in the physics engine. |

Implements **gazebo::physics::Link** (p. 559).

#### 10.181.3.28 virtual void gazebo::physics::SimbodyLink::SetForce ( const math::Vector3 & *_force* ) [virtual]

Set the force applied to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_force* | Force value. |

Implements **gazebo::physics::Link** (p. 559).

#### 10.181.3.29 virtual void gazebo::physics::SimbodyLink::SetGravityMode ( bool *_mode* ) [virtual]

Set whether gravity affects this body.

**Parameters**

| | | |
|---|---|---|
| in | *_mode* | True to enable gravity. |

Implements **gazebo::physics::Link** (p. 559).

#### 10.181.3.30 virtual void gazebo::physics::SimbodyLink::SetLinearDamping ( double *_damping* ) [virtual]

Set the linear damping factor.

**Parameters**

| | | |
|---|---|---|
| in | *_damping* | Linear damping factor. |

Implements **gazebo::physics::Link** (p. 560).

#### 10.181.3.31 virtual void gazebo::physics::SimbodyLink::SetLinearVel ( const math::Vector3 & *_vel* ) [virtual]

Set the linear velocity of the body.

**Parameters**

| | | |
|---|---|---|
| in | *_vel* | Linear velocity. |

Implements **gazebo::physics::Link** (p. 560).

#### 10.181.3.32 virtual void gazebo::physics::SimbodyLink::SetLinkStatic ( bool *_static* ) [virtual]

If the inboard body of this link is ground, simply lock the inboard joint to freeze it to ground.

Otherwise, add a weld constraint to simulate freeze to ground effect.

**Parameters**

| | | |
|---|---|---|
| in | *_static* | if true, freeze link to ground. Otherwise unfreeze link. |

Implements **gazebo::physics::Link** (p. 560).

#### 10.181.3.33 virtual void gazebo::physics::SimbodyLink::SetSelfCollide ( bool *_collide* ) [virtual]

Set whether this body will collide with others in the model.

**Parameters**

| | | |
|---|---|---|
| in | *_collid* | True to enable collisions. |

Implements **gazebo::physics::Link** (p. 561).

#### 10.181.3.34 virtual void gazebo::physics::SimbodyLink::SetTorque ( const **math::Vector3** & *_torque* ) [virtual]

Set the torque applied to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_torque* | Torque value. |

Implements **gazebo::physics::Link** (p. 561).

### 10.181.4 Member Data Documentation

#### 10.181.4.1 SimTK::MobilizedBody gazebo::physics::SimbodyLink::masterMobod

#### 10.181.4.2 bool gazebo::physics::SimbodyLink::mustBeBaseLink

: Force this link to be a base body, where its inboard body is the world with 6DOF.

#### 10.181.4.3 bool gazebo::physics::SimbodyLink::physicsInitialized

#### 10.181.4.4 std::vector<SimTK::MobilizedBody> gazebo::physics::SimbodyLink::slaveMobods

#### 10.181.4.5 std::vector<SimTK::Constraint::Weld> gazebo::physics::SimbodyLink::slaveWelds

The documentation for this class was generated from the following file:

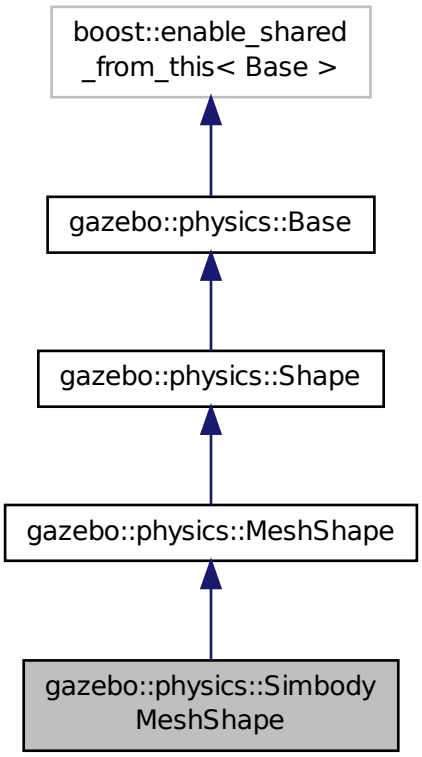
- **SimbodyLink.hh**

## 10.182 gazebo::physics::SimbodyMeshShape Class Reference

Triangle mesh collision.

`#include <SimbodyMeshShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyMeshShape:




### Public Member Functions

- **SimbodyMeshShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**SimbodyMeshShape** ()

  *Destructor.*

- virtual void **Load** (sdf::ElementPtr _sdf)
  *Load.*

### Protected Member Functions

- virtual void **Init** ()
  *Initialize the shape.*

### Additional Inherited Members

## 10.182.1 Detailed Description

Triangle mesh collision.

## 10.182.2 Constructor & Destructor Documentation

#### 10.182.2.1 gazebo::physics::SimbodyMeshShape::SimbodyMeshShape ( CollisionPtr *_parent* )

Constructor.

#### 10.182.2.2 virtual gazebo::physics::SimbodyMeshShape::~SimbodyMeshShape ( ) [virtual]

Destructor.

## 10.182.3 Member Function Documentation

#### 10.182.3.1 virtual void gazebo::physics::SimbodyMeshShape::Init ( ) [protected], [virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::MeshShape** (p. 623).

#### 10.182.3.2 virtual void gazebo::physics::SimbodyMeshShape::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load.

**Parameters**

| | | |
|---|---|---|
| in | *node* | Pointer to an SDF parameters |

Reimplemented from **gazebo::physics::Base** (p. 168).

The documentation for this class was generated from the following file:

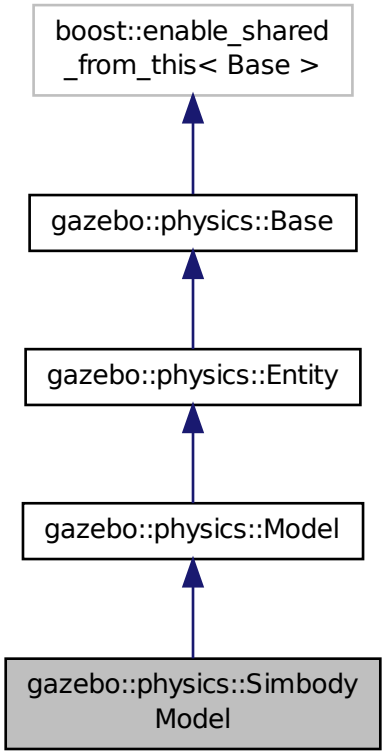
- **SimbodyMeshShape.hh**

## 10.183 gazebo::physics::SimbodyModel Class Reference

A model is a collection of links, joints, and plugins.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::SimbodyModel:




### Public Member Functions

- **SimbodyModel** (**BasePtr** _parent)

  *Constructor.*
- virtual ~**SimbodyModel** ()

  *Destructor.*
- virtual void **Init** ()

  *Initialize the model.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the model.*

### Additional Inherited Members

### 10.183.1 Detailed Description

A model is a collection of links, joints, and plugins.

### 10.183.2 Constructor & Destructor Documentation

#### 10.183.2.1 gazebo::physics::SimbodyModel::SimbodyModel ( BasePtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent object. |

#### 10.183.2.2 virtual gazebo::physics::SimbodyModel::~SimbodyModel ( ) [virtual]

Destructor.

### 10.183.3 Member Function Documentation

#### 10.183.3.1 virtual void gazebo::physics::SimbodyModel::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Model** (p. 633).

#### 10.183.3.2 virtual void gazebo::physics::SimbodyModel::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the model.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters to load from. |

Reimplemented from **gazebo::physics::Model** (p. 633).

The documentation for this class was generated from the following file:

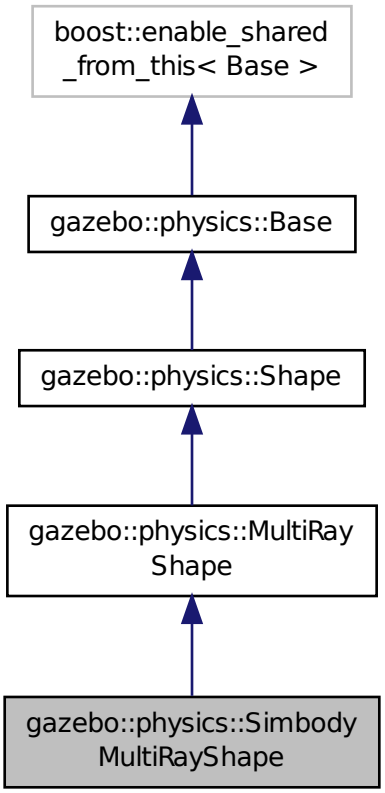
- **SimbodyModel.hh**

## 10.184 gazebo::physics::SimbodyMultiRayShape Class Reference

Simbody specific version of **MultiRayShape** (p. 664).

`#include <SimbodyMultiRayShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyMultiRayShape:




## Public Member Functions

- **SimbodyMultiRayShape** (**CollisionPtr parent**)

  *Constructor.*
- virtual ~**SimbodyMultiRayShape** ()

  *Destructor.*
- virtual void **UpdateRays** ()

  *Physics engine specific method for updating the rays.*

## Protected Member Functions

- virtual void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)

  *Add a ray to the collision.*

### Additional Inherited Members

### 10.184.1 Detailed Description

Simbody specific version of **MultiRayShape** (p. 664).

### 10.184.2 Constructor & Destructor Documentation

#### 10.184.2.1 gazebo::physics::SimbodyMultiRayShape::SimbodyMultiRayShape ( CollisionPtr *parent* )

Constructor.

#### 10.184.2.2 virtual gazebo::physics::SimbodyMultiRayShape::~SimbodyMultiRayShape ( ) [virtual]

Destructor.

### 10.184.3 Member Function Documentation

#### 10.184.3.1 virtual void gazebo::physics::SimbodyMultiRayShape::AddRay ( const math::Vector3 & *_start,* const math::Vector3 & *_end* ) [protected], [virtual]

Add a ray to the collision.

**Parameters**

| | | |
|---|---|---|
| in | *_start* | Start of the ray. |
| in | *_end* | End of the ray. |

Reimplemented from **gazebo::physics::MultiRayShape** (p. 667).

#### 10.184.3.2 virtual void gazebo::physics::SimbodyMultiRayShape::UpdateRays ( ) [virtual]

Physics engine specific method for updating the rays.

Implements **gazebo::physics::MultiRayShape** (p. 671).

The documentation for this class was generated from the following file:

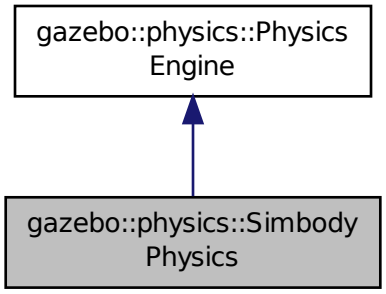
- **SimbodyMultiRayShape.hh**

## 10.185 gazebo::physics::SimbodyPhysics Class Reference

Simbody physics engine.

`#include <SimbodyPhysics.hh>`

Inheritance diagram for gazebo::physics::SimbodyPhysics:




## Public Member Functions

- **SimbodyPhysics** (**WorldPtr** _world)

  *Constructor.*
- virtual ~**SimbodyPhysics** ()

  *Destructor.*
- virtual **CollisionPtr CreateCollision** (const std::string &_type, **LinkPtr** _body)

  *Create a collision.*
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)

  *Create a new joint.*
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)

  *Create a new body.*
- virtual **ModelPtr CreateModel** (**BasePtr** _parent)

  *Create a new model.*
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)

  *Create a **physics::Shape** (p. 861) object.*
- virtual void **DebugPrint** () const

  *Debug print out of the physic engine state.*
- virtual void **Fini** ()

  *Finilize the physics engine.*
- SimTK::MultibodySystem * **GetDynamicsWorld** () const

  *Register a joint with the dynamics world.*
- virtual std::string **GetType** () const

  *Return the physics engine type (ode|bullet|dart|simbody).*
- virtual void **Init** ()

  *Initialize the physics engine.*
- virtual void **InitForThread** ()

  *Init the engine for threads.*
- void **InitModel** (const **physics::ModelPtr** _model)

*Add a **Model** (p. 624) to the Simbody system.*

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the physics engine.*
- virtual void **Reset** ()

  *Rest the physics engine.*
- virtual void **SetGravity** (const **gazebo::math::Vector3** &_gravity)

  *Set the gavity vector.*
- virtual void **SetSeed** (uint32_t _seed)

  *Set the random number seed for the physics engine.*
- virtual void **UpdateCollision** ()

  *Update the physics engine collision.*
- virtual void **UpdatePhysics** ()

  *Update the physics engine.*

## Static Public Member Functions

- static SimTK::Transform **GetPose** (sdf::ElementPtr _element)

  *If the given element contains a <pose> element, return it as a Transform.*
- static std::string **GetTypeString** (unsigned int _type)

  *Convert **Base::GetType()** (p. 167) to string, this is needed by the MultibodyGraphMaker.*
- static std::string **GetTypeString** (**physics::Base::EntityType** _type)

  *Convert **Base::GetType()** (p. 167) to string, this is needed by the MultibodyGraphMaker.*
- static SimTK::Transform **Pose2Transform** (const **math::Pose** &_pose)

  *Convert the given pose in x,y,z,thetax,thetay,thetaz format to a Simbody Transform.*
- static SimTK::Quaternion **QuadToQuad** (const **math::Quaternion** &_q)

  *Convert **gazebo::math::Quaternion** (p. 761) to SimTK::Quaternion.*
- static **math::Quaternion QuadToQuad** (const SimTK::Quaternion &_q)

  *Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 761).*
- static **math::Pose Transform2Pose** (const SimTK::Transform &_xAB)

  *Convert a Simbody transform to a pose in x,y,z, thetax,thetay,thetaz format.*
- static **math::Vector3 Vec3ToVector3** (const SimTK::Vec3 &_v)

  *Convert SimTK::Vec3 to **gazebo::math::Vector3** (p. 1091).*
- static SimTK::Vec3 **Vector3ToVec3** (const **math::Vector3** &_v)

  *Convert **gazebo::math::Vector3** (p. 1091) to SimTK::Vec3.*

## Public Attributes

- SimTK::CompliantContactSubsystem **contact**
- SimTK::Force::DiscreteForces **discreteForces**
- SimTK::GeneralForceSubsystem **forces**
- SimTK::Force::Gravity **gravity**
- SimTK::Integrator * **integ**
- SimTK::SimbodyMatterSubsystem **matter**
- bool **simbodyPhysicsInitialized**

  *true if initialized*
- bool **simbodyPhysicsStepped**
- SimTK::MultibodySystem **system**
- SimTK::ContactTrackerSubsystem **tracker**

### Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)

  *virtual callback for gztopic "~/physics".*
- virtual void **OnRequest** (ConstRequestPtr &_msg)

  *virtual callback for gztopic "~/request".*

### Additional Inherited Members

## 10.185.1 Detailed Description

Simbody physics engine.

## 10.185.2 Constructor & Destructor Documentation

#### 10.185.2.1 gazebo::physics::SimbodyPhysics::SimbodyPhysics ( WorldPtr *_world* )

Constructor.

#### 10.185.2.2 virtual gazebo::physics::SimbodyPhysics::~SimbodyPhysics ( ) [virtual]

Destructor.

## 10.185.3 Member Function Documentation

#### 10.185.3.1 virtual CollisionPtr gazebo::physics::SimbodyPhysics::CreateCollision ( const std::string & *_shapeType,* LinkPtr *_link* ) [virtual]

Create a collision.

**Parameters**

| | | |
|---|---|---|
| in | *_shapeType* | Type of collision to create. |
| in | *_link* | Parent link. |

Implements **gazebo::physics::PhysicsEngine** (p. 710).

#### 10.185.3.2 virtual JointPtr gazebo::physics::SimbodyPhysics::CreateJoint ( const std::string & *_type,* ModelPtr *_parent* ) [virtual]

Create a new joint.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | Type of joint to create. |
| in | *_parent* | **Model** (p. 624) parent. |

Implements **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.185.3.3 virtual LinkPtr gazebo::physics::SimbodyPhysics::CreateLink ( ModelPtr _parent ) [virtual]

Create a new body.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent model for the link. |

Implements **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.185.3.4 virtual ModelPtr gazebo::physics::SimbodyPhysics::CreateModel ( BasePtr _base ) [virtual]

Create a new model.

**Parameters**

| | | |
|---|---|---|
| in | *_base* | Boost shared pointer to a new model. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.185.3.5 virtual ShapePtr gazebo::physics::SimbodyPhysics::CreateShape ( const std::string & _shapeType, CollisionPtr _collision ) [virtual]

Create a **physics::Shape** (p. 861) object.

**Parameters**

| | | |
|---|---|---|
| in | *_shapeType* | Type of shape to create. |
| in | *_collision* | **Collision** (p. 220) parent. |

Implements **gazebo::physics::PhysicsEngine** (p. 711).

#### 10.185.3.6 virtual void gazebo::physics::SimbodyPhysics::DebugPrint ( ) const [virtual]

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 712).

#### 10.185.3.7 virtual void gazebo::physics::SimbodyPhysics::Fini ( ) [virtual]

Finilize the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 712).

#### 10.185.3.8 SimTK::MultibodySystem* gazebo::physics::SimbodyPhysics::GetDynamicsWorld ( ) const

Register a joint with the dynamics world.

#### 10.185.3.9 static SimTK::Transform gazebo::physics::SimbodyPhysics::GetPose ( sdf::ElementPtr _element ) [static]

If the given element contains a <pose> element, return it as a Transform.

Otherwise return the identity Transform. If there is more than one <pose> element, only the first one is processed.

#### 10.185.3.10 virtual std::string gazebo::physics::SimbodyPhysics::GetType ( ) const [virtual]

Return the physics engine type (ode|bullet|dart|simbody).

**Returns**

Type of the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 714).

#### 10.185.3.11 static std::string gazebo::physics::SimbodyPhysics::GetTypeString ( unsigned int *_type* ) [static]

Convert **Base::GetType()** (p. 167) to string, this is needed by the MultibodyGraphMaker.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | **Joint** (p. 496) type returned by **Joint::GetType()** (p. 167). |

**Returns**

a hard-coded string needed by the MultibodyGraphMaker.

#### 10.185.3.12 static std::string gazebo::physics::SimbodyPhysics::GetTypeString ( physics::Base::EntityType *_type* ) [static]

Convert **Base::GetType()** (p. 167) to string, this is needed by the MultibodyGraphMaker.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | **Joint** (p. 496) type returned by **Joint::GetType()** (p. 167). |

**Returns**

a hard-coded string needed by the MultibodyGraphMaker.

#### 10.185.3.13 virtual void gazebo::physics::SimbodyPhysics::Init ( ) [virtual]

Initialize the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 715).

#### 10.185.3.14 virtual void gazebo::physics::SimbodyPhysics::InitForThread ( ) [virtual]

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 715).

#### 10.185.3.15 void gazebo::physics::SimbodyPhysics::InitModel ( const physics::ModelPtr _model )

Add a **Model** (p. 624) to the Simbody system.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the model to add into Simbody. |

#### 10.185.3.16 virtual void gazebo::physics::SimbodyPhysics::Load ( sdf::ElementPtr _sdf ) [virtual]

Load the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Pointer to the SDF parameters. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 715).

#### 10.185.3.17 virtual void gazebo::physics::SimbodyPhysics::OnPhysicsMsg ( ConstPhysicsPtr & _msg ) [protected], [virtual]

virtual callback for gztopic "~/physics".

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Physics message. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 716).

#### 10.185.3.18 virtual void gazebo::physics::SimbodyPhysics::OnRequest ( ConstRequestPtr & _msg ) [protected], [virtual]

virtual callback for gztopic "~/request".

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Request message. |

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 716).

#### 10.185.3.19 static SimTK::Transform gazebo::physics::SimbodyPhysics::Pose2Transform ( const math::Pose & _pose ) [static]

Convert the given pose in x,y,z,thetax,thetay,thetaz format to a Simbody Transform.

The rotation angles are interpreted as a body-fixed sequence, meaning we rotation about x, then about the new y, then about the now twice-rotated z.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | Gazeb's **math::Pose** (p. 734) object |

**Returns**

Simbody's SimTK::Transform object

#### 10.185.3.20 static SimTK::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad ( const math::Quaternion & _q ) [static]

Convert **gazebo::math::Quaternion** (p. 761) to SimTK::Quaternion.

**Parameters**

| | | |
|---|---|---|
| in | *_q* | Gazeb's **math::Quaternion** (p. 761) object |

**Returns**

Simbody's SimTK::Quaternion object

#### 10.185.3.21 static math::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad ( const SimTK::Quaternion & _q ) [static]

Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 761).

**Parameters**

| | | |
|---|---|---|
| in | *_q* | Simbody's SimTK::Quaternion object |

**Returns**

Gazeb's **math::Quaternion** (p. 761) object

#### 10.185.3.22 virtual void gazebo::physics::SimbodyPhysics::Reset ( ) [virtual]

Rest the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 716).

#### 10.185.3.23 virtual void gazebo::physics::SimbodyPhysics::SetGravity ( const gazebo::math::Vector3 & _gravity ) [virtual]

Set the gavity vector.

**Parameters**

| | | |
|---|---|---|
| in | *_gravity* | New gravity vector. |

Implements **gazebo::physics::PhysicsEngine** (p. 717).

#### 10.185.3.24 virtual void gazebo::physics::SimbodyPhysics::SetSeed ( uint32_t _seed ) [virtual]

Set the random number seed for the physics engine.

**Parameters**

| | | |
|---|---|---|
| in | *_seed* | The random number seed. |

Implements **gazebo::physics::PhysicsEngine** (p. 718).

#### 10.185.3.25 static math::Pose gazebo::physics::SimbodyPhysics::Transform2Pose ( const SimTK::Transform & *_xAB* ) [static]

Convert a Simbody transform to a pose in x,y,z, thetax,thetay,thetaz format.

**Parameters**

| | | |
|---|---|---|
| in | *_xAB* | Simbody's SimTK::Transform object |

**Returns**

Gazeb's **math::Pose** (p. 734) object

#### 10.185.3.26 virtual void gazebo::physics::SimbodyPhysics::UpdateCollision ( ) [virtual]

Update the physics engine collision.

Implements **gazebo::physics::PhysicsEngine** (p. 719).

#### 10.185.3.27 virtual void gazebo::physics::SimbodyPhysics::UpdatePhysics ( ) [virtual]

Update the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 719).

#### 10.185.3.28 static math::Vector3 gazebo::physics::SimbodyPhysics::Vec3ToVector3 ( const SimTK::Vec3 & *_v* ) [static]

Convert SimTK::Vec3 to **gazebo::math::Vector3** (p. 1091).

**Parameters**

| | | |
|---|---|---|
| in | *_v* | Simbody's SimTK::Vec3 object |

**Returns**

Gazeb's **math::Vector3** (p. 1091) object

#### 10.185.3.29 static SimTK::Vec3 gazebo::physics::SimbodyPhysics::Vector3ToVec3 ( const math::Vector3 & *_v* ) [static]

Convert **gazebo::math::Vector3** (p. 1091) to SimTK::Vec3.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | Gazeb's **math::Vector3** (p. 1091) object |

**Returns**

Simbody's SimTK::Vec3 object

### 10.185.4 Member Data Documentation

#### 10.185.4.1 SimTK::CompliantContactSubsystem gazebo::physics::SimbodyPhysics::contact

#### 10.185.4.2 SimTK::Force::DiscreteForces gazebo::physics::SimbodyPhysics::discreteForces

#### 10.185.4.3 SimTK::GeneralForceSubsystem gazebo::physics::SimbodyPhysics::forces

#### 10.185.4.4 SimTK::Force::Gravity gazebo::physics::SimbodyPhysics::gravity

#### 10.185.4.5 SimTK:: Integrator* gazebo::physics::SimbodyPhysics::integ

#### 10.185.4.6 SimTK::SimbodyMatterSubsystem gazebo::physics::SimbodyPhysics::matter

#### 10.185.4.7 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsInitialized

true if initialized

#### 10.185.4.8 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsStepped

#### 10.185.4.9 SimTK::MultibodySystem gazebo::physics::SimbodyPhysics::system

#### 10.185.4.10 SimTK::ContactTrackerSubsystem gazebo::physics::SimbodyPhysics::tracker

The documentation for this class was generated from the following file:

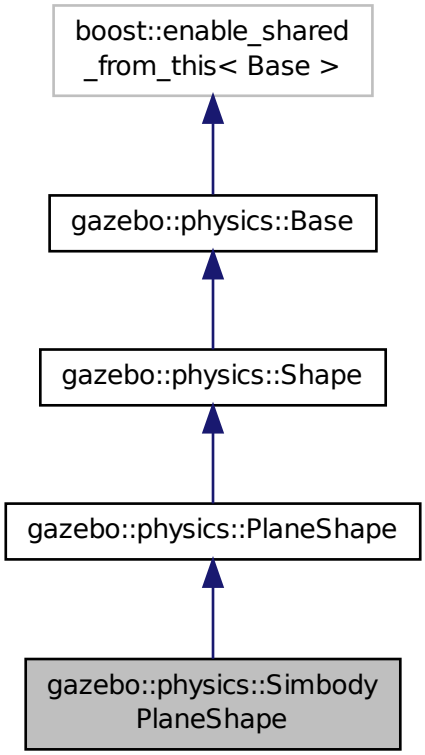
- **SimbodyPhysics.hh**

## 10.186 gazebo::physics::SimbodyPlaneShape Class Reference

Simbody collision for an infinite plane.

`#include <SimbodyPlaneShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyPlaneShape:




## Public Member Functions

- **SimbodyPlaneShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**SimbodyPlaneShape** ()

  *Destructor.*
- virtual void **CreatePlane** ()

  *Create the plane.*
- virtual void **SetAltitude** (const **math::Vector3** &_pos)

  *Set the altitude of the plane.*

## Additional Inherited Members

### 10.186.1 Detailed Description

Simbody collision for an infinite plane.

### 10.186.2 Constructor & Destructor Documentation

#### 10.186.2.1 gazebo::physics::SimbodyPlaneShape::SimbodyPlaneShape ( CollisionPtr _parent )

Constructor.

#### 10.186.2.2 virtual gazebo::physics::SimbodyPlaneShape::~SimbodyPlaneShape ( ) [virtual]

Destructor.

### 10.186.3 Member Function Documentation

#### 10.186.3.1 virtual void gazebo::physics::SimbodyPlaneShape::CreatePlane ( ) [virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 730).

#### 10.186.3.2 virtual void gazebo::physics::SimbodyPlaneShape::SetAltitude ( const math::Vector3 & _pos ) [virtual]

Set the altitude of the plane.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | Position of the plane. |

Reimplemented from **gazebo::physics::PlaneShape** (p. 731).

The documentation for this class was generated from the following file:

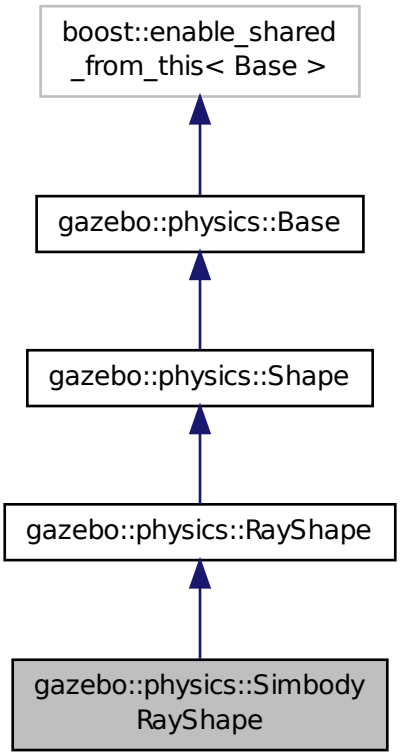
- **SimbodyPlaneShape.hh**

## 10.187 gazebo::physics::SimbodyRayShape Class Reference

Ray shape for simbody.

`#include <SimbodyRayShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyRayShape:




## Public Member Functions

- **SimbodyRayShape** (**PhysicsEnginePtr** _physicsEngine)

  *Constructor.*
- **SimbodyRayShape** (**CollisionPtr** _collision)

  *Constructor.*
- virtual ~**SimbodyRayShape** ()

  *Destructor.*
- virtual void **GetIntersection** (double &_dist, std::string &_entity)

  *Get the nearest intersection.*
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)

  *Set the ray based on starting and ending points relative to the body.*
- virtual void **Update** ()

  *Update the ray collision.*

### Additional Inherited Members

### 10.187.1 Detailed Description

Ray shape for simbody.

### 10.187.2 Constructor & Destructor Documentation

#### 10.187.2.1 gazebo::physics::SimbodyRayShape::SimbodyRayShape ( PhysicsEnginePtr *_physicsEngine* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_physicsEngine* | Pointer to the physics engine. |

#### 10.187.2.2 gazebo::physics::SimbodyRayShape::SimbodyRayShape ( CollisionPtr *_collision* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_collision* | **Collision** (p. 220) the ray is attached to. |

#### 10.187.2.3 virtual gazebo::physics::SimbodyRayShape::~SimbodyRayShape ( ) [virtual]

Destructor.

### 10.187.3 Member Function Documentation

#### 10.187.3.1 virtual void gazebo::physics::SimbodyRayShape::GetIntersection ( double & *_dist,* std::string & *_entity* ) [virtual]

Get the nearest intersection.

**Parameters**

| | | |
|---|---|---|
| out | *_dist* | Distance to the intersection. |
| out | *_entity* | Name of the entity the ray intersected with. |

Implements **gazebo::physics::RayShape** (p. 789).

#### 10.187.3.2 virtual void gazebo::physics::SimbodyRayShape::SetPoints ( const math::Vector3 & *_posStart,* const math::Vector3 & *_posEnd* ) [virtual]

Set the ray based on starting and ending points relative to the body.

**Parameters**

| | | |
|---|---|---|
| in | *_posStart* | Start position, relative the body. |
| in | *_posEnd* | End position, relative to the body. |

Reimplemented from **gazebo::physics::RayShape** (p. 790).

#### 10.187.3.3 virtual void gazebo::physics::SimbodyRayShape::Update ( ) [virtual]

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 791).

The documentation for this class was generated from the following file:

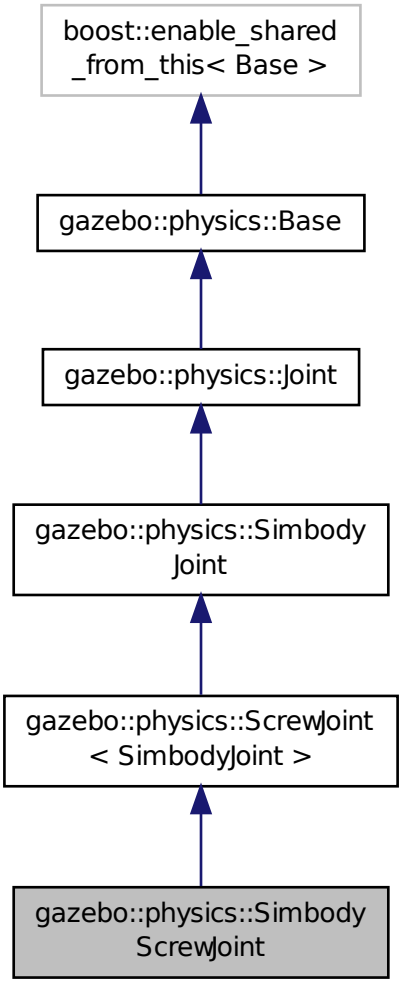
- **SimbodyRayShape.hh**

## 10.188 gazebo::physics::SimbodyScrewJoint Class Reference

A screw joint.

`#include <SimbodyScrewJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyScrewJoint:




## Public Member Functions

- **SimbodyScrewJoint** (SimTK::MultibodySystem *_world, **BasePtr** _parent)

  *Constructor.*
- virtual ~**SimbodyScrewJoint** ()

  *Destructor.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*
- virtual **math::Angle GetHighStop** (int _index)

*Get the high stop of an axis(index).*

- virtual **math::Angle GetLowStop** (int _index)

  *Get the low stop of an axis(index).*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetThreadPitch** (unsigned int)

  *Get screw joint thread pitch.*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, double _damping)

  *Set the joint damping.*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*
- virtual void **SetMaxForce** (int _index, double _t)

  *Set the max allowed force of an axis(index).*
- virtual void **SetThreadPitch** (int _index, double _threadPitch)

  *Set screw joint thread pitch.*
- virtual void **SetVelocity** (int _index, double _angle)

  *Set the velocity of an axis(index).*

## Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **ScrewJoint** (p. 832).*
- virtual void **SetForceImpl** (int _index, double _force)

  *Set the force applied to this **physics::Joint** (p. 496).*

## Additional Inherited Members

### 10.188.1 Detailed Description

A screw joint.

### 10.188.2 Constructor & Destructor Documentation

#### 10.188.2.1 gazebo::physics::SimbodyScrewJoint::SimbodyScrewJoint ( SimTK::MultibodySystem * *_world,* BasePtr *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to the Simbody world. |
| in | *_parent* | Parent of the screw joint. |

#### 10.188.2.2 virtual gazebo::physics::SimbodyScrewJoint::~SimbodyScrewJoint ( ) [virtual]

Destructor.

### 10.188.3 Member Function Documentation

#### 10.188.3.1 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.188.3.2 virtual math::Vector3 gazebo::physics::SimbodyScrewJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.188.3.3 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetHighStop ( int _index ) [virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 506).

#### 10.188.3.4 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetLowStop ( int _index ) [virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.188.3.5 virtual double gazebo::physics::SimbodyScrewJoint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.188.3.6 virtual double gazebo::physics::SimbodyScrewJoint::GetThreadPitch ( unsigned int _index ) [virtual]

Get screw joint thread pitch.

This must be implemented in a child class

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

_threadPitch Thread pitch value.

Implements **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 834).

#### 10.188.3.7 virtual double gazebo::physics::SimbodyScrewJoint::GetVelocity ( int _index ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.188.3.8 virtual void gazebo::physics::SimbodyScrewJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 510).

#### 10.188.3.9 virtual void gazebo::physics::SimbodyScrewJoint::Load ( sdf::ElementPtr *_sdf* ) [protected], [virtual]

Load a **ScrewJoint** (p. 832).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF value to load from |

Reimplemented from **gazebo::physics::ScrewJoint**< **SimbodyJoint** > (p. 834).

#### 10.188.3.10 virtual void gazebo::physics::SimbodyScrewJoint::SetAxis ( int *_index,* const **math::Vector3** & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.188.3.11 virtual void gazebo::physics::SimbodyScrewJoint::SetDamping ( int *_index,* double *_damping* ) [virtual]

Set the joint damping.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set, currently ignored, to be implemented. |
| in | *_damping* | Damping value for the axis. |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.188.3.12 virtual void gazebo::physics::SimbodyScrewJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only. |

Implements **gazebo::physics::SimbodyJoint** (p. 899).

#### 10.188.3.13 virtual void gazebo::physics::SimbodyScrewJoint::SetHighStop ( int *_index*, const **math::Angle** & *_angle* ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.188.3.14 virtual void gazebo::physics::SimbodyScrewJoint::SetLowStop ( int *_index*, const **math::Angle** & *_angle* ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.188.3.15 virtual void gazebo::physics::SimbodyScrewJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.188.3.16 virtual void gazebo::physics::SimbodyScrewJoint::SetThreadPitch ( int *_index*, double *_threadPitch* ) [virtual]

Set screw joint thread pitch.

This must be implemented in a child class

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_threadPitch* | Thread pitch value. |

Implements **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 835).

#### 10.188.3.17 virtual void gazebo::physics::SimbodyScrewJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

The documentation for this class was generated from the following file:

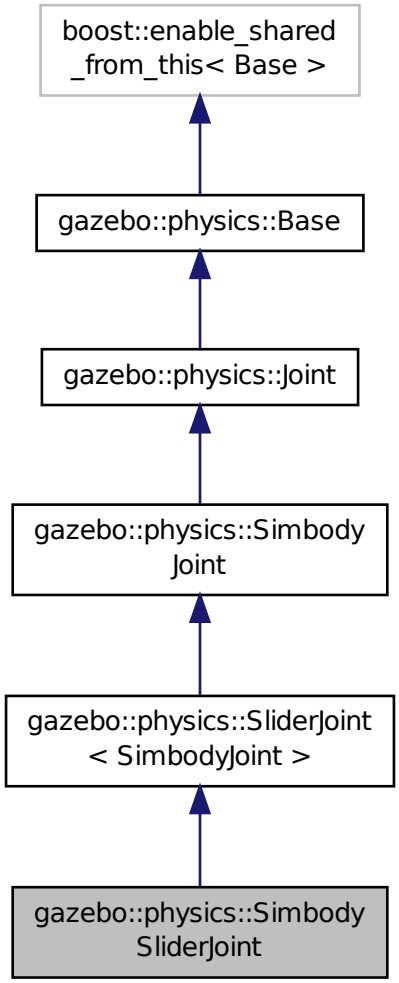
- **SimbodyScrewJoint.hh**

## 10.189 gazebo::physics::SimbodySliderJoint Class Reference

A slider joint.

`#include <SimbodySliderJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodySliderJoint:




## Public Member Functions

- **SimbodySliderJoint** (SimTK::MultibodySystem ∗**world**, **BasePtr** _parent)

  *Constructor.*
- virtual ∼**SimbodySliderJoint** ()

  *Destructor.*
- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*
- virtual **math::Angle GetHighStop** (int _index)

*Get the high stop of an axis(index).*

- virtual **math::Angle GetLowStop** (int _index)

  *Get the low stop of an axis(index).*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, const double _damping)

  *Set the joint damping.*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*
- virtual void **SetMaxForce** (int _index, double _t)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _rate)

  *Set the velocity of an axis(index).*

## Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **SliderJoint** (p. 973).*
- virtual void **SetForceImpl** (int _index, double _force)

  *Set the force applied to this **physics::Joint** (p. 496).*

## Additional Inherited Members

### 10.189.1 Detailed Description

A slider joint.

### 10.189.2 Constructor & Destructor Documentation

#### 10.189.2.1 gazebo::physics::SimbodySliderJoint::SimbodySliderJoint ( SimTK::MultibodySystem * *world,* BasePtr *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to the Simbody world. |
| in | *_parent* | Parent of the screw joint. |

#### 10.189.2.2 virtual gazebo::physics::SimbodySliderJoint::~SimbodySliderJoint ( ) [virtual]

Destructor.

### 10.189.3 Member Function Documentation

#### 10.189.3.1 virtual math::Angle gazebo::physics::SimbodySliderJoint::GetAngleImpl ( int _index ) const [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.189.3.2 virtual math::Vector3 gazebo::physics::SimbodySliderJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.189.3.3 virtual math::Angle gazebo::physics::SimbodySliderJoint::GetHighStop ( int _index ) [virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 506).

#### 10.189.3.4 virtual math::Angle gazebo::physics::SimbodySliderJoint::GetLowStop ( int _index ) [virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.189.3.5 virtual double gazebo::physics::SimbodySliderJoint::GetMaxForce ( int *_index* ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.189.3.6 virtual double gazebo::physics::SimbodySliderJoint::GetVelocity ( int *_index* ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.189.3.7 virtual void gazebo::physics::SimbodySliderJoint::Load ( sdf::ElementPtr *_sdf* ) [protected], [virtual]

Load a **SliderJoint** (p. 973).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from |

Reimplemented from **gazebo::physics::SliderJoint**< **SimbodyJoint** > (p. 975).

#### 10.189.3.8 virtual void gazebo::physics::SimbodySliderJoint::SetAxis ( int _index, const math::Vector3 & _axis ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.189.3.9 virtual void gazebo::physics::SimbodySliderJoint::SetDamping ( int _index, const double _damping ) [virtual]

Set the joint damping.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set, currently ignored, to be implemented. |
| in | *_damping* | Damping value for the axis. |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.189.3.10 virtual void gazebo::physics::SimbodySliderJoint::SetForceImpl ( int _index, double _force ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only. |

Implements **gazebo::physics::SimbodyJoint** (p. 899).

#### 10.189.3.11 virtual void gazebo::physics::SimbodySliderJoint::SetHighStop ( int _index, const math::Angle & _angle ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.189.3.12 virtual void gazebo::physics::SimbodySliderJoint::SetLowStop ( int *_index*, const math::Angle & *_angle* ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.189.3.13 virtual void gazebo::physics::SimbodySliderJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.189.3.14 virtual void gazebo::physics::SimbodySliderJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

The documentation for this class was generated from the following file:

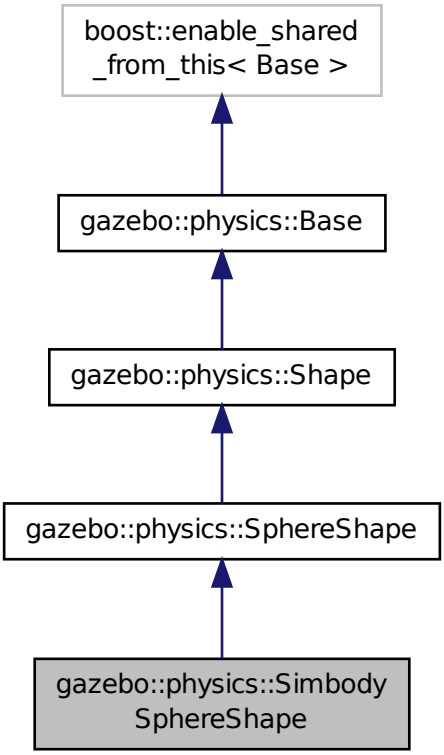
- **SimbodySliderJoint.hh**

## 10.190 gazebo::physics::SimbodySphereShape Class Reference

Simbody sphere collision.

`#include <SimbodySphereShape.hh>`

Inheritance diagram for gazebo::physics::SimbodySphereShape:




## Public Member Functions

- **SimbodySphereShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**SimbodySphereShape** ()

  *Destructor.*
- virtual void **SetRadius** (double _radius)

  *Set the size.*

## Additional Inherited Members

### 10.190.1 Detailed Description

Simbody sphere collision.

### 10.190.2 Constructor & Destructor Documentation

#### 10.190.2.1 gazebo::physics::SimbodySphereShape::SimbodySphereShape ( CollisionPtr _parent ) [inline]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | **Collision** (p. 220) parent pointer |

#### 10.190.2.2 virtual gazebo::physics::SimbodySphereShape::~SimbodySphereShape ( ) [inline], [virtual]

Destructor.

### 10.190.3 Member Function Documentation

#### 10.190.3.1 virtual void gazebo::physics::SimbodySphereShape::SetRadius ( double _radius ) [inline], [virtual]

Set the size.

**Parameters**

| | | |
|---|---|---|
| in | *_radius* | Radius of the sphere. |

Reimplemented from **gazebo::physics::SphereShape** (p. 988).

References gazebo::physics::Shape::collisionParent, gazebo::math::equal(), gzerr, gzwarn, and gazebo::physics::SphereShape::SetRadius().

The documentation for this class was generated from the following file:

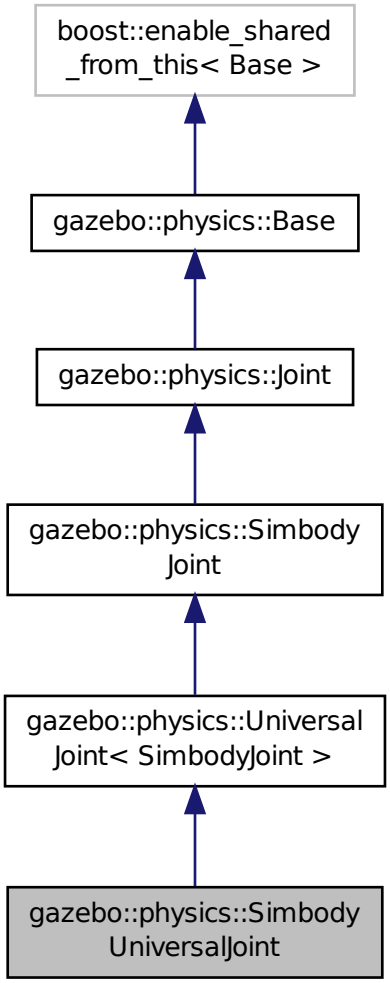
- **SimbodySphereShape.hh**

## 10.191 gazebo::physics::SimbodyUniversalJoint Class Reference

A simbody universal joint class.

`#include <SimbodyUniversalJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyUniversalJoint:




## Public Member Functions

- **SimbodyUniversalJoint** (SimTK::MultibodySystem *_world, **BasePtr** _parent)

  *Constructor.*
- virtual ~**SimbodyUniversalJoint** ()

  *Destuctor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor point.*
- virtual **math::Vector3 GetAxis** (int _index) const
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  *Get the axis of rotation in global cooridnate frame.*

- virtual **math::Angle GetHighStop** (int _index)

  *Get the high stop of an axis(index).*
- virtual **math::Angle GetLowStop** (int _index)

  *Get the low stop of an axis(index).*
- virtual double **GetMaxForce** (int _index)

  *Get the max allowed force of an axis(index).*
- virtual double **GetVelocity** (int _index) const

  *Get the rotation rate of an axis(index)*
- virtual void **Init** ()

  *Initialize a joint.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **UniversalJoint** (*p. *1064).*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  *Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, double _damping)

  *Set the joint damping.*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  *Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  *Set the low stop of an axis(index).*
- virtual void **SetMaxForce** (int _index, double _t)

  *Set the max allowed force of an axis(index).*
- virtual void **SetVelocity** (int _index, double _angle)

  *Set the velocity of an axis(index).*

### Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const

  *Get the angle of an axis helper function.*
- virtual void **SetForceImpl** (int _index, double _torque)

  *Set the force applied to this **physics::Joint** (*p. *496).*

### Additional Inherited Members

## 10.191.1 Detailed Description

A simbody universal joint class.

## 10.191.2 Constructor & Destructor Documentation

### 10.191.2.1 gazebo::physics::SimbodyUniversalJoint::SimbodyUniversalJoint ( SimTK::MultibodySystem * *_world,* BasePtr *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to the Simbody world. |
| in | *_parent* | Parent of the screw joint. |

#### 10.191.2.2 virtual gazebo::physics::SimbodyUniversalJoint::~SimbodyUniversalJoint ( ) [virtual]

Destuctor.

### 10.191.3 Member Function Documentation

#### 10.191.3.1 virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAnchor ( int _index ) const [virtual]

Get the anchor point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Anchor value for the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 894).

#### 10.191.3.2 virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetAngleImpl ( int _index ) const [protected], [virtual]

Get the angle of an axis helper function.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 503).

#### 10.191.3.3 virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAxis ( int _index ) const [virtual]

#### 10.191.3.4 virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetGlobalAxis ( int _index ) const [virtual]

Get the axis of rotation in global cooridnate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to get. |

**Returns**

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 505).

#### 10.191.3.5 virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetHighStop ( int _index ) [virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 506).

#### 10.191.3.6 virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetLowStop ( int _index ) [virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.191.3.7 virtual double gazebo::physics::SimbodyUniversalJoint::GetMaxForce ( int _index ) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The maximum force.

Implements **gazebo::physics::Joint** (p. 508).

#### 10.191.3.8 virtual double gazebo::physics::SimbodyUniversalJoint::GetVelocity ( int _index ) const [virtual]

Get the rotation rate of an axis(index)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |

**Returns**

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 509).

#### 10.191.3.9 virtual void gazebo::physics::SimbodyUniversalJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 510).

#### 10.191.3.10 virtual void gazebo::physics::SimbodyUniversalJoint::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load a **UniversalJoint** (p. 1064).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented from **gazebo::physics::UniversalJoint**< **SimbodyJoint** > (p. 1065).

#### 10.191.3.11 virtual void gazebo::physics::SimbodyUniversalJoint::SetAxis ( int *_index*, const **math::Vector3** & *_axis* ) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set. |
| in | *_axis* | Vector in local joint frame of axis direction (must have length greater than zero). |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.191.3.12 virtual void gazebo::physics::SimbodyUniversalJoint::SetDamping ( int *_index*, double *_damping* ) [virtual]

Set the joint damping.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis to set, currently ignored, to be implemented. |
| in | *_damping* | Damping value for the axis. |

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 898).

#### 10.191.3.13 virtual void gazebo::physics::SimbodyUniversalJoint::SetForceImpl ( int *_index*, double *_force* ) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 496).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 496)).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only. |

Implements **gazebo::physics::SimbodyJoint** (p. 899).

#### 10.191.3.14 virtual void gazebo::physics::SimbodyUniversalJoint::SetHighStop ( int *_index*, const **math::Angle** & *_angle* ) [virtual]

Set the high stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | High stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.191.3.15 virtual void gazebo::physics::SimbodyUniversalJoint::SetLowStop ( int *_index*, const **math::Angle** & *_angle* ) [virtual]

Set the low stop of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_angle* | Low stop angle. |

Reimplemented from **gazebo::physics::Joint** (p. 513).

#### 10.191.3.16 virtual void gazebo::physics::SimbodyUniversalJoint::SetMaxForce ( int *_index*, double *_force* ) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_force* | Maximum force that can be applied to the axis. |

Implements **gazebo::physics::Joint** (p. 513).

#### 10.191.3.17 virtual void gazebo::physics::SimbodyUniversalJoint::SetVelocity ( int *_index*, double *_vel* ) [virtual]

Set the velocity of an axis(index).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. |
| in | *_vel* | Velocity. |

Implements **gazebo::physics::Joint** (p. 514).

The documentation for this class was generated from the following file:

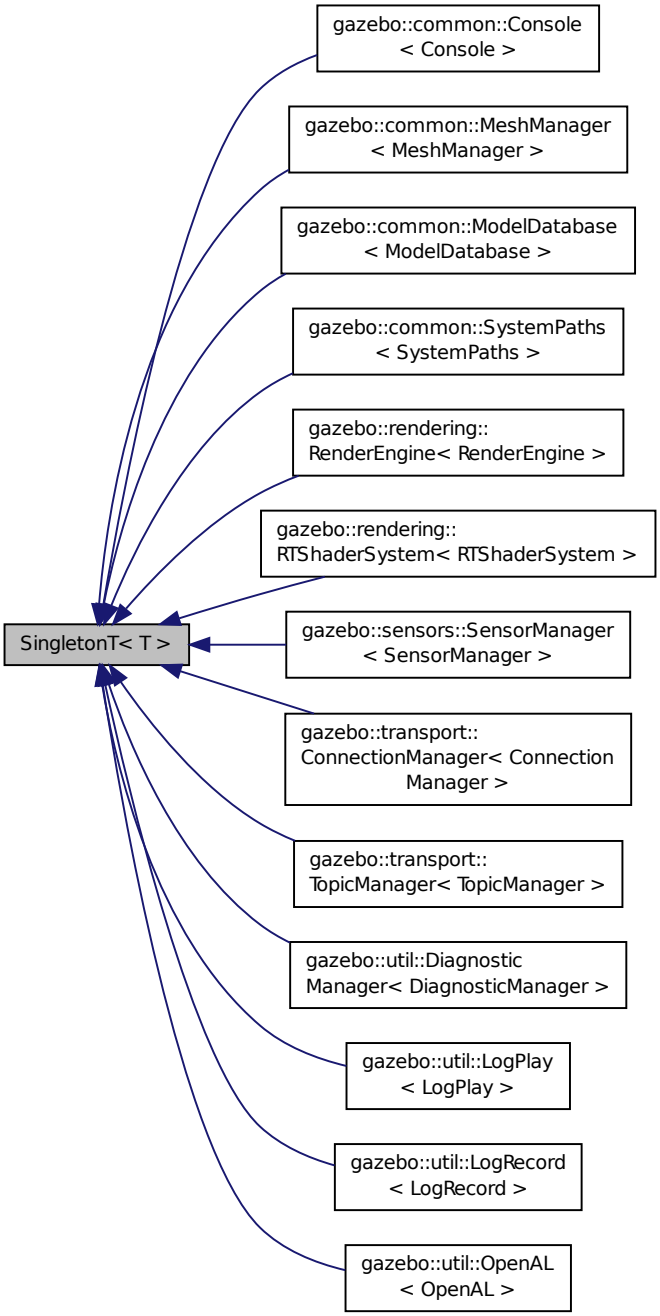
- **SimbodyUniversalJoint.hh**

## 10.192 SingletonT< T > Class Template Reference

Singleton template class.

`#include <common/common.hh>`

Inheritance diagram for SingletonT< T >:




## Static Public Member Functions

- static T * **Instance** ()

*Get an instance of the singleton.*

### Protected Member Functions

- **SingletonT** ()
  *Constructor.*
- virtual ~**SingletonT** ()
  *Destructor.*

### 10.192.1 Detailed Description

**template<class T>class SingletonT< T >**

Singleton template class.

### 10.192.2 Constructor & Destructor Documentation

#### 10.192.2.1 template<class T> SingletonT< T >::SingletonT ( ) [inline], [protected]

Constructor.

#### 10.192.2.2 template<class T> virtual SingletonT< T >::~SingletonT ( ) [inline], [protected], [virtual]

Destructor.

### 10.192.3 Member Function Documentation

#### 10.192.3.1 template<class T> static T* SingletonT< T >::Instance ( ) [inline], [static]

Get an instance of the singleton.

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::transport::Node::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), and gazebo::transport::Node::Subscribe().

The documentation for this class was generated from the following file:

- **SingletonT.hh**

## 10.193 gazebo::common::Skeleton Class Reference

A skeleton.

`#include <common/common.hh>`

### Public Member Functions

- **Skeleton** ()
  *Constructor.*

- **Skeleton** (**SkeletonNode** ∗_root)

  *Constructor.*
- virtual ∼**Skeleton** ()

  *Destructor.*
- void **AddAnimation** (**SkeletonAnimation** ∗_anim)

  *Add an animation.*
- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)

  *Add a new weight to a node (bone)*
- **SkeletonAnimation** ∗ **GetAnimation** (const unsigned int _i)

  *Find animation.*
- **math::Matrix4 GetBindShapeTransform** ()

  *Return bind pose skeletal transform.*
- **SkeletonNode** ∗ **GetNodeByHandle** (unsigned int _handle)

  *Find or create node with handle.*
- **SkeletonNode** ∗ **GetNodeById** (std::string _id)

  *Find node by index.*
- **SkeletonNode** ∗ **GetNodeByName** (std::string _name)

  *Find a node.*
- **NodeMap GetNodes** ()

  *Get a copy or the node dictionary.*
- unsigned int **GetNumAnimations** ()

  *Returns the number of animations.*
- unsigned int **GetNumJoints** ()

  *Returns the number of joints.*
- unsigned int **GetNumNodes** ()

  *Returns the node count.*
- unsigned int **GetNumVertNodeWeights** (unsigned int _vertex)

  *Returns the number of bone weights for a vertex.*
- **SkeletonNode** ∗ **GetRootNode** ()

  *Return the root.*
- std::pair< std::string, double > **GetVertNodeWeight** (unsigned int _v, unsigned int _i)

  *Weight of a bone for a vertex.*
- void **PrintTransforms** ()

  *Outputs the transforms to std::err stream.*
- void **Scale** (double _scale)

  *Scale all nodes, transforms and animation data.*
- void **SetBindShapeTransform** (**math::Matrix4** _trans)

  *Set the bind pose skeletal transform.*
- void **SetNumVertAttached** (unsigned int _vertices)

  *Resizes the raw node weight array.*
- void **SetRootNode** (**SkeletonNode** ∗_node)

  *Change the root node.*

## Protected Member Functions

- void **BuildNodeMap** ()

  *Initializes the hande numbers for each node in the map using breadth first traversal.*

### Protected Attributes

- std::vector< **SkeletonAnimation** * > **anims**

  *the array of animations*
- **math::Matrix4 bindShapeTransform**

  *the bind pose skeletal transform*
- **NodeMap nodes**

  *The dictionary of nodes, indexed by name.*
- **RawNodeWeights rawNW**

  *the node weight table*
- **SkeletonNode** * **root**

  *the root node*

## 10.193.1 Detailed Description

A skeleton.

## 10.193.2 Constructor & Destructor Documentation

#### 10.193.2.1 gazebo::common::Skeleton::Skeleton ( )

Constructor.

#### 10.193.2.2 gazebo::common::Skeleton::Skeleton ( SkeletonNode * *_root* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_root* | node |

#### 10.193.2.3 virtual gazebo::common::Skeleton::~Skeleton ( ) [virtual]

Destructor.

## 10.193.3 Member Function Documentation

#### 10.193.3.1 void gazebo::common::Skeleton::AddAnimation ( SkeletonAnimation * *_anim* )

Add an animation.

The skeleton does not take ownership of the animation

**Parameters**

| | | |
|---|---|---|
| in | *_anim* | the animation to add |

#### 10.193.3.2 void gazebo::common::Skeleton::AddVertNodeWeight ( unsigned int _vertex, std::string _node, double _weight )

Add a new weight to a node (bone)

**Parameters**

| | | |
|---|---|---|
| in | *_vertex* | index of the vertex |
| in | *_node* | name of the bone |
| in | *_weight* | the new weight (range 0 to 1) |

#### 10.193.3.3 void gazebo::common::Skeleton::BuildNodeMap ( ) [protected]

Initializes the hande numbers for each node in the map using breadth first traversal.

#### 10.193.3.4 SkeletonAnimation* gazebo::common::Skeleton::GetAnimation ( const unsigned int _i )

Find animation.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | the animation index |

**Returns**

the animation, or NULL if _i is out of bounds

#### 10.193.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ( )

Return bind pose skeletal transform.

**Returns**

a matrix

#### 10.193.3.6 SkeletonNode* gazebo::common::Skeleton::GetNodeByHandle ( unsigned int _handle )

Find or create node with handle.

**Parameters**

| | | |
|---|---|---|
| in | *_handle* | |

**Returns**

the node. A new node is created if it didn't exist

#### 10.193.3.7 SkeletonNode* gazebo::common::Skeleton::GetNodeById ( std::string _id )

Find node by index.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | the index |

**Returns**

the node, or NULL if not found

#### 10.193.3.8 SkeletonNode* gazebo::common::Skeleton::GetNodeByName ( std::string *_name* )

Find a node.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the node to look for |

**Returns**

the node, or NULL if not found

#### 10.193.3.9 NodeMap gazebo::common::Skeleton::GetNodes ( )

Get a copy or the node dictionary.

#### 10.193.3.10 unsigned int gazebo::common::Skeleton::GetNumAnimations ( )

Returns the number of animations.

**Returns**

the count

#### 10.193.3.11 unsigned int gazebo::common::Skeleton::GetNumJoints ( )

Returns the number of joints.

**Returns**

the count

#### 10.193.3.12 unsigned int gazebo::common::Skeleton::GetNumNodes ( )

Returns the node count.

**Returns**

the count

#### 10.193.3.13 unsigned int gazebo::common::Skeleton::GetNumVertNodeWeights ( unsigned int *_vertex* )

Returns the number of bone weights for a vertex.

**Parameters**

| | | |
|---|---|---|
| in | *_vertex* | the index of the vertex |

**Returns**

the count

#### 10.193.3.14 SkeletonNode* gazebo::common::Skeleton::GetRootNode ( )

Return the root.

**Returns**

the root

#### 10.193.3.15 std::pair<std::string, double> gazebo::common::Skeleton::GetVertNodeWeight ( unsigned int *_v*, unsigned int *_i* )

Weight of a bone for a vertex.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the index of the vertex |
| in | *_i* | the index of the weight for that vertex |

**Returns**

a pair containing the name of the node and the weight

#### 10.193.3.16 void gazebo::common::Skeleton::PrintTransforms ( )

Outputs the transforms to std::err stream.

#### 10.193.3.17 void gazebo::common::Skeleton::Scale ( double *_scale* )

Scale all nodes, transforms and animation data.

**Parameters**

| | | |
|---|---|---|
| in | *the* | scaling factor |

#### 10.193.3.18 void gazebo::common::Skeleton::SetBindShapeTransform ( math::Matrix4 *_trans* )

Set the bind pose skeletal transform.

**Parameters**

| | | |
|---|---|---|
| in | *_trans* | the transform |

#### 10.193.3.19 void gazebo::common::Skeleton::SetNumVertAttached ( unsigned int *_vertices* )

Resizes the raw node weight array.

**Parameters**

| | | |
|---|---|---|
| in | *_vertices* | the new size |

#### 10.193.3.20 void gazebo::common::Skeleton::SetRootNode ( SkeletonNode * *_node* )

Change the root node.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | the new node |

### 10.193.4 Member Data Documentation

#### 10.193.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anims [protected]

the array of animations

#### 10.193.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

#### 10.193.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

#### 10.193.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

#### 10.193.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

The documentation for this class was generated from the following file:

- **Skeleton.hh**

## 10.194 gazebo::common::SkeletonAnimation Class Reference

**Skeleton** (p. 953) animation.

`#include <SkeletonAnimation.hh>`

### Public Member Functions

- **SkeletonAnimation** (const std::string &_name)

  *The Constructor.*
- ∼**SkeletonAnimation** ()

  *The destructor.*
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Matrix4** _mat)

  *Adds or replaces a named key frame at a specific time.*
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Pose** _pose)

  *Adds or replaces a named key frame at a specific time.*
- double **GetLength** () const

  *Returns the duration of the animations.*
- std::string **GetName** () const

  *Returns the name.*
- unsigned int **GetNodeCount** () const

  *Returns the number of animation nodes.*
- **math::Matrix4 GetNodePoseAt** (const std::string &_node, const double _time, const bool _loop=true)

  *Returns the key frame transformation for a named animation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.*
- std::map< std::string, **math::Matrix4** > **GetPoseAt** (const double _time, const bool _loop=true) const

  *Returns a dictionary of transformations indexed by name at a specific time if a node does not exist at that specific time (with tolerance of 1e-6 sec), the transformation is interpolated.*
- std::map< std::string, **math::Matrix4** > **GetPoseAtX** (const double _x, const std::string &_node, const bool _loop=true) const

  *Returns a dictionary of transformations indexed by name where a named node transformation's translational value along the X axis is equal to _x.*
- bool **HasNode** (const std::string &_node) const

  *Looks for a node with a specific name in the animations.*
- void **Scale** (const double _scale)

  *Scales every animation in the animations list.*
- void **SetName** (const std::string &_name)

  *Changes the name.*

### Protected Attributes

- std::map< std::string, **NodeAnimation** ∗ > **animations**

  *a dictionary of node animations*
- double **length**

  *the duration of the longest animation*
- std::string **name**

  *the node name*

### 10.194.1 Detailed Description

**Skeleton** (p. 953) animation.

### 10.194.2 Constructor & Destructor Documentation

#### 10.194.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation ( const std::string & *_name* )

The Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the animation |

#### 10.194.2.2 gazebo::common::SkeletonAnimation::~SkeletonAnimation ( )

The destructor.

Clears the list without destroying the animations

### 10.194.3 Member Function Documentation

#### 10.194.3.1 void gazebo::common::SkeletonAnimation::AddKeyFrame ( const std::string & *_node,* const double *_time,* const math::Matrix4 *_mat* )

Adds or replaces a named key frame at a specific time.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | the name of the new or existing node |
| in | *_time* | the time |
| in | *_mat* | the key frame transformation |

#### 10.194.3.2 void gazebo::common::SkeletonAnimation::AddKeyFrame ( const std::string & *_node,* const double *_time,* const math::Pose *_pose* )

Adds or replaces a named key frame at a specific time.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | the name of the new or existing node |
| in | *_time* | the time |
| in | *_pose* | the key frame transformation as a **math::Pose** (p. 734) |

#### 10.194.3.3 double gazebo::common::SkeletonAnimation::GetLength ( ) const

Returns the duration of the animations.

**Returns**

the duration in seconds

#### 10.194.3.4 std::string gazebo::common::SkeletonAnimation::GetName ( ) const

Returns the name.

**Returns**

the name

#### 10.194.3.5 unsigned int gazebo::common::SkeletonAnimation::GetNodeCount ( ) const

Returns the number of animation nodes.

**Returns**

the count

#### 10.194.3.6 math::Matrix4 gazebo::common::SkeletonAnimation::GetNodePoseAt ( const std::string & *_node*, const double *_time*, const bool *_loop* = `true` )

Returns the key frame transformation for a named animation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

**Parameters**

| | | |
|---|---|---|
| `in` | *_node* | the name of the animation node |
| `in` | *_time* | the time |
| `in` | *_loop* | when true, the time is divided by the duration (see GetLength) |

**Returns**

the transformation

#### 10.194.3.7 std::map<std::string, math::Matrix4> gazebo::common::SkeletonAnimation::GetPoseAt ( const double *_time*, const bool *_loop* = `true` ) const

Returns a dictionary of transformations indexed by name at a specific time if a node does not exist at that specific time (with tolerance of 1e-6 sec), the transformation is interpolated.

**Parameters**

| | | |
|---|---|---|
| `in` | *_time* | the time |
| `in` | *_loop* | when true, the time is divided by the duration (see GetLength) |

**Returns**

the transformation for every node

#### 10.194.3.8 std::map<std::string, math::Matrix4> gazebo::common::SkeletonAnimation::GetPoseAtX ( const double _x, const std::string & _node, const bool _loop = true ) const

Returns a dictionary of transformations indexed by name where a named node transformation's translational value along the X axis is equal to _x.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | the value along x. You must ensure that _x is within a valid range. |
| in | *_node* | the name of the animation node |
| in | *_loop* | when true, the time is divided by the duration (see GetLength) |

#### 10.194.3.9 bool gazebo::common::SkeletonAnimation::HasNode ( const std::string & _node ) const

Looks for a node with a specific name in the animations.

**Parameters**

| | | |
|---|---|---|
| in | *_node* | the name of the node |

**Returns**

true if the node exits

#### 10.194.3.10 void gazebo::common::SkeletonAnimation::Scale ( const double _scale )

Scales every animation in the animations list.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | the scaling factor |

#### 10.194.3.11 void gazebo::common::SkeletonAnimation::SetName ( const std::string & _name )

Changes the name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the new name |

### 10.194.4 Member Data Documentation

#### 10.194.4.1 std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations [protected]

a dictionary of node animations

#### 10.194.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

#### 10.194.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

## 10.195 gazebo::common::SkeletonNode Class Reference

A skeleton node.

`#include <common/common.hh>`

### Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }
  *enumeration of node types*

### Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)
  *Constructor.*
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)
  *Constructor.*
- virtual ~**SkeletonNode** ()
  *Destructor.*
- void **AddChild** (**SkeletonNode** *_child)
  *Add a new child.*
- void **AddRawTransform** (**NodeTransform** _t)
  *Add a raw transform.*
- **SkeletonNode** * **GetChild** (unsigned int _index)
  *Find a child by index.*
- **SkeletonNode** * **GetChildById** (std::string _id)
  *Get child by string id.*
- **SkeletonNode** * **GetChildByName** (std::string _name)
  *Get child by name.*
- unsigned int **GetChildCount** ()
  *Returns the children count.*
- unsigned int **GetHandle** ()
  *Get the handle index.*
- std::string **GetId** ()
  *Returns the index.*

- **math::Matrix4 GetInverseBindTransform** ()

  *Retrieve the inverse of the bind pose skeletal transform.*
- **math::Matrix4 GetModelTransform** ()

  *Retrieve the model transform.*
- std::string **GetName** ()

  *Returns the name.*
- unsigned int **GetNumRawTrans** ()

  *Return the raw transformations count.*
- **SkeletonNode** ∗ **GetParent** ()

  *Returns the parent node.*
- **NodeTransform GetRawTransform** (unsigned int _i)

  *Find a raw transformation.*
- std::vector< **NodeTransform** > **GetRawTransforms** ()

  *Retrieve the raw transformations.*
- **math::Matrix4 GetTransform** ()

  *Get transform relative to parent.*
- std::vector< **NodeTransform** > **GetTransforms** ()

  *Returns a copy of the array of transformations.*
- bool **IsJoint** ()

  *Is a joint query.*
- bool **IsRootNode** ()

  *Queries wether a node has no parent parent.*
- void **Reset** (bool _resetChildren)

  *Reset the transformation to the initial transformation.*
- void **SetHandle** (unsigned int _h)

  *Assign a handle number.*
- void **SetId** (std::string _id)

  *Change the id string.*
- void **SetInitialTransform** (**math::Matrix4** _tras)

  *Sets the initial transformation.*
- void **SetInverseBindTransform** (**math::Matrix4** _invBM)

  *Assign the inverse of the bind pose skeletal transform.*
- void **SetModelTransform** (**math::Matrix4** _trans, bool _updateChildren=true)

  *Set the model transformation.*
- void **SetName** (std::string _name)

  *Change the name.*
- void **SetParent** (**SkeletonNode** ∗_parent)

  *Set the parent node.*
- void **SetTransform** (**math::Matrix4** _trans, bool _updateChildren=true)

  *Set a transformation.*
- void **SetType** (**SkeletonNodeType** _type)

  *Change the skeleton node type.*
- void **UpdateChildrenTransforms** ()

  *Apply model transformations in order for each node in the tree.*

### Protected Attributes

- std::vector< **SkeletonNode** * > **children**

  *the children nodes*
- unsigned int **handle**

  *handle index number*
- std::string **id**

  *a string identifier*
- **math::Matrix4 initialTransform**

  *the initial transformation*
- **math::Matrix4 invBindTransform**

  *the inverse of the bind pose skeletal transform*
- **math::Matrix4 modelTransform**

  *the model transformation*
- std::string **name**

  *the name of the skeletal node*
- **SkeletonNode** * **parent**

  *the parent node*
- std::vector< **NodeTransform** > **rawTransforms**

  *the raw transformation*
- **math::Matrix4 transform**

  *the transform*
- **SkeletonNodeType type**

  *the type fo node*

## 10.195.1 Detailed Description

A skeleton node.

## 10.195.2 Member Enumeration Documentation

### 10.195.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

**Enumerator**

***NODE***

***JOINT***

## 10.195.3 Constructor & Destructor Documentation

### 10.195.3.1 gazebo::common::SkeletonNode::SkeletonNode ( SkeletonNode * *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | The parent node |

#### 10.195.3.2 gazebo::common::SkeletonNode::SkeletonNode ( SkeletonNode * _parent, std::string _name, std::string _id, SkeletonNodeType _type = JOINT )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | the parent node |
| in | *_name* | name of node |
| in | *_id* | Id of node |
| in | *_type* | The type of this node |

#### 10.195.3.3 virtual gazebo::common::SkeletonNode::~SkeletonNode ( ) [virtual]

Destructor.

### 10.195.4 Member Function Documentation

#### 10.195.4.1 void gazebo::common::SkeletonNode::AddChild ( SkeletonNode * _child )

Add a new child.

**Parameters**

| | | |
|---|---|---|
| in | *_child* | a child |

#### 10.195.4.2 void gazebo::common::SkeletonNode::AddRawTransform ( NodeTransform _t )

Add a raw transform.

**Parameters**

| | | |
|---|---|---|
| in | *_t* | the transform |

#### 10.195.4.3 SkeletonNode* gazebo::common::SkeletonNode::GetChild ( unsigned int _index )

Find a child by index.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | the index |

**Returns**

the child skeleton. NO BOUNDS CHECKING

#### 10.195.4.4 SkeletonNode* gazebo::common::SkeletonNode::GetChildById ( std::string _id )

Get child by string id.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | the string id |

**Returns**

the child skeleton or NULL if not found

#### 10.195.4.5 SkeletonNode* gazebo::common::SkeletonNode::GetChildByName ( std::string *_name* )

Get child by name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the name of the child skeleton |

**Returns**

the skeleton, or NULL if not found

#### 10.195.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ( )

Returns the children count.

**Returns**

the count

#### 10.195.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ( )

Get the handle index.

**Returns**

the handle index

#### 10.195.4.8 std::string gazebo::common::SkeletonNode::GetId ( )

Returns the index.

**Returns**

the id string

#### 10.195.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ( )

Retrieve the inverse of the bind pose skeletal transform.

**Returns**

the transform

#### 10.195.4.10 math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ( )

Retrieve the model transform.

**Returns**

the transform

#### 10.195.4.11 std::string gazebo::common::SkeletonNode::GetName ( )

Returns the name.

**Returns**

the name

#### 10.195.4.12 unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ( )

Return the raw transformations count.

**Returns**

the count

#### 10.195.4.13 SkeletonNode* gazebo::common::SkeletonNode::GetParent ( )

Returns the parent node.

**Returns**

the parent

#### 10.195.4.14 NodeTransform gazebo::common::SkeletonNode::GetRawTransform ( unsigned int _i )

Find a raw transformation.

**Parameters**

| | | |
|---|---|---|
| in | _i | the index of the transformation |

**Returns**

the node transform. NO BOUNDS CHECKING PERFORMED

#### 10.195.4.15 std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ( )

Retrieve the raw transformations.

**Returns**

an array of transformations

#### 10.195.4.16 math::Matrix4 gazebo::common::SkeletonNode::GetTransform ( )

Get transform relative to parent.

#### 10.195.4.17 std::vector<NodeTransform> gazebo::common::SkeletonNode::GetTransforms ( )

Returns a copy of the array of transformations.

**Returns**

the array of transform (These are the same as the raw trans)

#### 10.195.4.18 bool gazebo::common::SkeletonNode::IsJoint ( )

Is a joint query.

**Returns**

true if the skeleton type is a joint, false otherwise

#### 10.195.4.19 bool gazebo::common::SkeletonNode::IsRootNode ( )

Queries wether a node has no parent parent.

**Returns**

true if the node has no parent, fasle otherwise

#### 10.195.4.20 void gazebo::common::SkeletonNode::Reset ( bool *_resetChildren* )

Reset the transformation to the initial transformation.

**Parameters**

| | | |
|---|---|---|
| in | *_resetChildren* | when true, performs the operation for every node in the tree |

#### 10.195.4.21 void gazebo::common::SkeletonNode::SetHandle ( unsigned int *_h* )

Assign a handle number.

**Parameters**

| | | |
|---|---|---|
| in | *_h* | the handle |

#### 10.195.4.22 void gazebo::common::SkeletonNode::SetId ( std::string _id )

Change the id string.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | the new id string |

#### 10.195.4.23 void gazebo::common::SkeletonNode::SetInitialTransform ( math::Matrix4 _tras )

Sets the initial transformation.

**Parameters**

| | | |
|---|---|---|
| in | *_tras* | the transfromation matrix |

#### 10.195.4.24 void gazebo::common::SkeletonNode::SetInverseBindTransform ( math::Matrix4 _invBM )

Assign the inverse of the bind pose skeletal transform.

**Parameters**

| | | |
|---|---|---|
| in | *_invBM* | the transform |

#### 10.195.4.25 void gazebo::common::SkeletonNode::SetModelTransform ( math::Matrix4 _trans, bool _updateChildren = true )

Set the model transformation.

**Parameters**

| | | |
|---|---|---|
| in | *_trans* | the transformation |
| in | *_updateChildren* | when true the UpdateChildrenTransforms operation is performed |

#### 10.195.4.26 void gazebo::common::SkeletonNode::SetName ( std::string _name )

Change the name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | the new name |

#### 10.195.4.27 void gazebo::common::SkeletonNode::SetParent ( SkeletonNode * _parent )

Set the parent node.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | the new parent |

#### 10.195.4.28 void gazebo::common::SkeletonNode::SetTransform ( math::Matrix4 *_trans*, bool *_updateChildren* = `true` )

Set a transformation.

**Parameters**

| | | |
|---|---|---|
| in | *_trans* | the transformation |
| in | *_updateChildren* | when true the UpdateChildrenTransforms operation is performed |

#### 10.195.4.29 void gazebo::common::SkeletonNode::SetType ( SkeletonNodeType *_type* )

Change the skeleton node type.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | the new type |

#### 10.195.4.30 void gazebo::common::SkeletonNode::UpdateChildrenTransforms ( )

Apply model transformations in order for each node in the tree.

### 10.195.5 Member Data Documentation

#### 10.195.5.1 std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children [protected]

the children nodes

#### 10.195.5.2 unsigned int gazebo::common::SkeletonNode::handle [protected]

handle index number

#### 10.195.5.3 std::string gazebo::common::SkeletonNode::id [protected]

a string identifier

#### 10.195.5.4 math::Matrix4 gazebo::common::SkeletonNode::initialTransform [protected]

the initial transformation

#### 10.195.5.5 math::Matrix4 gazebo::common::SkeletonNode::invBindTransform [protected]

the inverse of the bind pose skeletal transform

#### 10.195.5.6 math::Matrix4 gazebo::common::SkeletonNode::modelTransform [protected]

the model transformation

#### 10.195.5.7 std::string gazebo::common::SkeletonNode::name [protected]

the name of the skeletal node

#### 10.195.5.8 SkeletonNode* gazebo::common::SkeletonNode::parent [protected]

the parent node

#### 10.195.5.9 std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms [protected]

the raw transformation

#### 10.195.5.10 math::Matrix4 gazebo::common::SkeletonNode::transform [protected]

the transform

#### 10.195.5.11 SkeletonNodeType gazebo::common::SkeletonNode::type [protected]

the type fo node

The documentation for this class was generated from the following file:

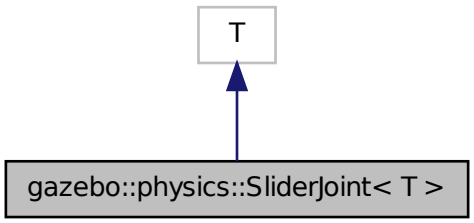
- **Skeleton.hh**

## 10.196 gazebo::physics::SliderJoint< T > Class Template Reference

A slider joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::SliderJoint< T >:




### Public Member Functions

- **SliderJoint** (**BasePtr** _parent)

*Constructor.*

- virtual ~**SliderJoint** ()

  *Destructor.*
- virtual **math::Vector3 GetAnchor** (int _index) const

  *Get the anchor.*
- virtual unsigned int **GetAngleCount** () const

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **SliderJoint** (p. 973).*
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)

  *Set the anchor.*

## Protected Attributes

- **math::Vector3 fakeAnchor**

  *The anchor value is not used internally.*

### 10.196.1 Detailed Description

**template<class T>class gazebo::physics::SliderJoint< T >**

A slider joint.

### 10.196.2 Constructor & Destructor Documentation

#### 10.196.2.1 template<class T> gazebo::physics::SliderJoint< T >::SliderJoint ( BasePtr *_parent* ) [inline], [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent of the joint. |

#### 10.196.2.2 template<class T> virtual gazebo::physics::SliderJoint< T >::~SliderJoint ( ) [inline], [virtual]

Destructor.

### 10.196.3 Member Function Documentation

#### 10.196.3.1 template<class T > math::Vector3 gazebo::physics::SliderJoint< T >::GetAnchor ( int *_index* ) const [virtual]

Get the anchor.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. Not used. |

**Returns**

Anchor for the joint.

Reimplemented in **gazebo::physics::DARTSliderJoint** (p. 347).

#### 10.196.3.2 template<class T> virtual unsigned int gazebo::physics::SliderJoint< T >::GetAngleCount ( ) const [inline], [virtual]

#### 10.196.3.3 template<class T> virtual void gazebo::physics::SliderJoint< T >::Load ( sdf::ElementPtr *_sdf* ) [inline], [virtual]

Load a **SliderJoint** (p. 973).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from |

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 940), and **gazebo::physics::DARTSliderJoint** (p. 348).

#### 10.196.3.4 template<class T > void gazebo::physics::SliderJoint< T >::SetAnchor ( int *_index*, const math::Vector3 & *_anchor* ) [virtual]

Set the anchor.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the axis. Not used. |
| in | *_anchor* | Anchor for the axis. |

### 10.196.4 Member Data Documentation

#### 10.196.4.1 template<class T> math::Vector3 gazebo::physics::SliderJoint< T >::fakeAnchor [protected]

The anchor value is not used internally.

The documentation for this class was generated from the following file:

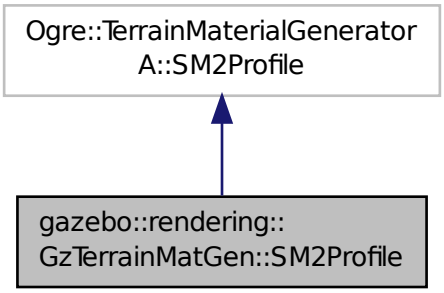
- **SliderJoint.hh**

## 10.197 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference

Shader model 2 profile target.

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile:




## Classes

- class **ShaderHelperCg**

  *Keeping the CG shader for reference.*
- class **ShaderHelperGLSL**

  *Utility class to help with generating shaders for GLSL.*

## Public Member Functions

- **SM2Profile** (Ogre::TerrainMaterialGenerator ∗_parent, const Ogre::String &_name, const Ogre::String &_desc)

  *Constructor.*
- virtual ∼**SM2Profile** ()

  *Destructor.*
- Ogre::MaterialPtr **generate** (const Ogre::Terrain ∗_terrain)
- Ogre::MaterialPtr **generateForCompositeMap** (const Ogre::Terrain ∗_terrain)
- void **UpdateParams** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain)
- void **UpdateParamsForCompositeMap** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain)

## Protected Member Functions

- virtual void **addTechnique** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain, TechniqueType _tt)

### 10.197.1 Detailed Description

Shader model 2 profile target.

### 10.197.2 Constructor & Destructor Documentation

#### 10.197.2.1 gazebo::rendering::GzTerrainMatGen::SM2Profile::SM2Profile ( Ogre::TerrainMaterialGenerator * *_parent,* const Ogre::String & *_name,* const Ogre::String & *_desc* )

Constructor.

#### 10.197.2.2 virtual gazebo::rendering::GzTerrainMatGen::SM2Profile::~SM2Profile ( ) [virtual]

Destructor.

### 10.197.3 Member Function Documentation

#### 10.197.3.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::addTechnique ( const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt* ) [protected], [virtual]

#### 10.197.3.2 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generate ( const Ogre::Terrain * *_terrain* )

#### 10.197.3.3 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generateForCompositeMap ( const Ogre::Terrain * *_terrain* )

#### 10.197.3.4 void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParams ( const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain* )

#### 10.197.3.5 void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParamsForCompositeMap ( const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain* )

The documentation for this class was generated from the following file:

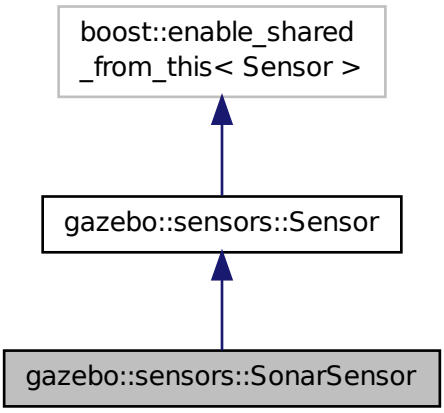
- **Heightmap.hh**

## 10.198 gazebo::sensors::SonarSensor Class Reference

**Sensor** (p. 837) with sonar cone.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::SonarSensor:




## Public Member Functions

- **SonarSensor** ()

  *Constructor.*
- virtual ~**SonarSensor** ()

  *Destructor.*
- template<typename T >
  **event::ConnectionPtr ConnectUpdate** (T _subscriber)

  *Connect a to the new update signal.*
- void **DisconnectUpdate** (**event::ConnectionPtr** &_conn)

  *Disconnect from the update signal.*
- double **GetRadius** () const

  *Get the radius of the sonar cone at maximum range.*
- double **GetRange** ()

  *Get detected range for a sonar.*
- double **GetRangeMax** () const

  *Get the minimum range of the sonar.*
- double **GetRangeMin** () const

  *Get the minimum range of the sonar.*
- virtual std::string **GetTopic** () const

  *Returns the topic name as set in SDF.*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual bool **IsActive** ()

  *Returns true if sensor generation is active.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

### Protected Member Functions

- virtual void **Fini** ()

  *Finalize the sensor.*
- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Protected Attributes

- **event::EventT**< void(msgs::SonarStamped)> **update**

  *Update event.*

## 10.198.1 Detailed Description

**Sensor** (p. 837) with sonar cone.

This sensor uses a cone .

## 10.198.2 Constructor & Destructor Documentation

### 10.198.2.1 gazebo::sensors::SonarSensor::SonarSensor ( )

Constructor.

### 10.198.2.2 virtual gazebo::sensors::SonarSensor::~SonarSensor ( ) [virtual]

Destructor.

## 10.198.3 Member Function Documentation

### 10.198.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::SonarSensor::ConnectUpdate ( T _subscriber ) [inline]

Connect a to the new update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_subscriber* | Callback function. |

**Returns**

The connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect(), and update.

### 10.198.3.2 void gazebo::sensors::SonarSensor::DisconnectUpdate ( event::ConnectionPtr & _conn ) [inline]

Disconnect from the update signal.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | Connection to remove. |

References gazebo::event::EventT< T >::Disconnect(), and update.

#### 10.198.3.3 virtual void gazebo::sensors::SonarSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

#### 10.198.3.4 double gazebo::sensors::SonarSensor::GetRadius ( ) const

Get the radius of the sonar cone at maximum range.

**Returns**

The radisu of the sonar cone at max range.

#### 10.198.3.5 double gazebo::sensors::SonarSensor::GetRange ( )

Get detected range for a sonar.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

**Returns**

Returns DBL_MAX for no detection.

#### 10.198.3.6 double gazebo::sensors::SonarSensor::GetRangeMax ( ) const

Get the minimum range of the sonar.

**Returns**

The sonar's maximum range.

#### 10.198.3.7 double gazebo::sensors::SonarSensor::GetRangeMin ( ) const

Get the minimum range of the sonar.

**Returns**

The sonar's minimum range.

#### 10.198.3.8 virtual std::string gazebo::sensors::SonarSensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

**Returns**

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 843).

#### 10.198.3.9 virtual void gazebo::sensors::SonarSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.198.3.10 virtual bool gazebo::sensors::SonarSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

**Returns**

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

#### 10.198.3.11 virtual void gazebo::sensors::SonarSensor::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

#### 10.198.3.12 virtual void gazebo::sensors::SonarSensor::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

### 10.198.4 Member Data Documentation

#### 10.198.4.1 event::EventT<void(msgs::SonarStamped)> gazebo::sensors::SonarSensor::update [protected]

Update event.

Referenced by ConnectUpdate(), and DisconnectUpdate().

The documentation for this class was generated from the following file:

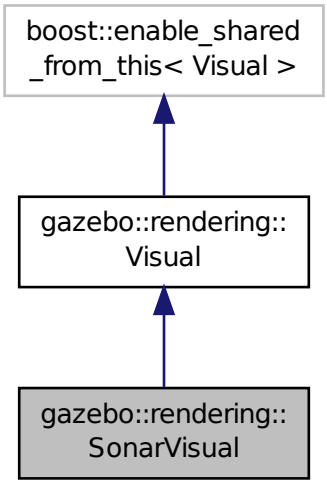
- **SonarSensor.hh**

## 10.199 gazebo::rendering::SonarVisual Class Reference

Visualization for sonar data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::SonarVisual:




### Public Member Functions

- **SonarVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  *Constructor.*
- virtual ~**SonarVisual** ()

  *Destructor.*
- virtual void **Load** ()

  *Load the visual with default parameters.*

### Additional Inherited Members

### 10.199.1 Detailed Description

Visualization for sonar data.

### 10.199.2 Constructor & Destructor Documentation

#### 10.199.2.1 gazebo::rendering::SonarVisual::SonarVisual ( const std::string & *_name,* VisualPtr *_vis,* const std::string & *_topicName* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual. |
| in | *_vis* | Pointer to the parent **Visual** (p. 1121). |
| in | *_topicName* | Name of the topic that has sonar data. |

#### 10.199.2.2 virtual gazebo::rendering::SonarVisual::~SonarVisual ( ) [virtual]

Destructor.

### 10.199.3 Member Function Documentation

#### 10.199.3.1 virtual void gazebo::rendering::SonarVisual::Load ( ) [virtual]

Load the visual with default parameters.

Reimplemented from **gazebo::rendering::Visual** (p. 1135).

The documentation for this class was generated from the following file:

- **SonarVisual.hh**

## 10.200 Joint_TEST::SpawnJointOptions Class Reference

Class to hold parameters for spawning joints.

`#include <Joint_TEST.hh>`

### Public Member Functions

- **SpawnJointOptions** ()

  *Constructor.*
- ~**SpawnJointOptions** ()

  *Destructor.*

### Public Attributes

- **math::Vector3 axis**

  *Axis value for spawned joint.*
- **math::Pose childLinkPose**

  *Child link pose for spawned model.*
- **math::Pose jointPose**

  *Joint pose for spawned joint.*
- **math::Pose modelPose**

  *Model pose for spawned model.*
- bool **noLinkPose**

  *Flag to disable including link pose per issue #978.*
- **math::Pose parentLinkPose**

  *Parent link pose for spawned model.*
- std::string **type**

  *Type of joint to create.*
- **common::Time wait**

  *Length of time to wait for model to spawn in order to return Joint pointer.*
- bool **worldChild**

  *Flag to set child link to the world.*
- bool **worldParent**

  *Flag to set parent link to the world.*

## 10.200.1 Detailed Description

Class to hold parameters for spawning joints.

## 10.200.2 Constructor & Destructor Documentation

### 10.200.2.1 Joint_TEST::SpawnJointOptions::SpawnJointOptions ( ) [inline]

Constructor.

### 10.200.2.2 Joint_TEST::SpawnJointOptions::~SpawnJointOptions ( ) [inline]

Destructor.

## 10.200.3 Member Data Documentation

### 10.200.3.1 math::Vector3 Joint_TEST::SpawnJointOptions::axis

Axis value for spawned joint.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.2 math::Pose Joint_TEST::SpawnJointOptions::childLinkPose

Child link pose for spawned model.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.3 math::Pose Joint_TEST::SpawnJointOptions::jointPose

Joint pose for spawned joint.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.4 math::Pose Joint_TEST::SpawnJointOptions::modelPose

Model pose for spawned model.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.5 bool Joint_TEST::SpawnJointOptions::noLinkPose

Flag to disable including link pose per issue #978.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.6 math::Pose Joint_TEST::SpawnJointOptions::parentLinkPose

Parent link pose for spawned model.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.7 std::string Joint_TEST::SpawnJointOptions::type

Type of joint to create.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.8 common::Time Joint_TEST::SpawnJointOptions::wait

Length of time to wait for model to spawn in order to return Joint pointer.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.9 bool Joint_TEST::SpawnJointOptions::worldChild

Flag to set child link to the world.

Referenced by Joint_TEST::SpawnJoint().

#### 10.200.3.10 bool Joint_TEST::SpawnJointOptions::worldParent

Flag to set parent link to the world.

Referenced by Joint_TEST::SpawnJoint().

The documentation for this class was generated from the following file:

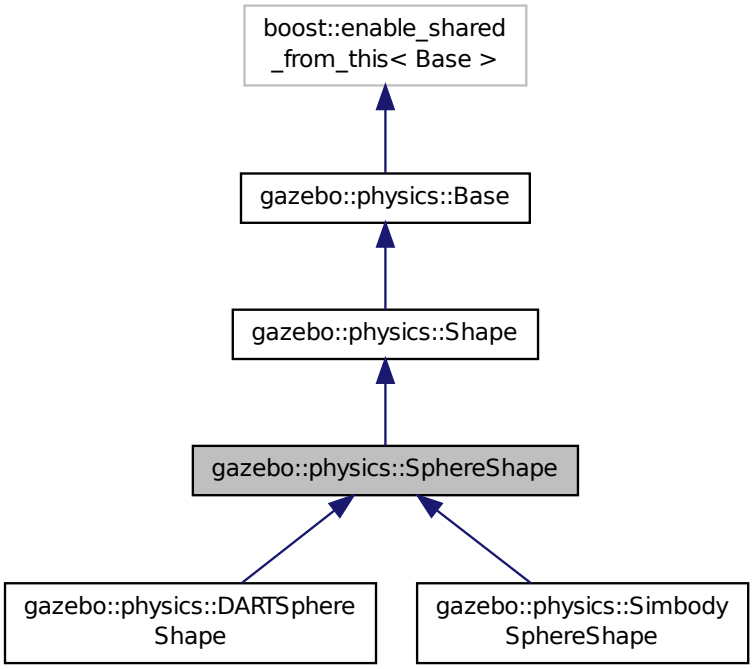
- **Joint_TEST.hh**

## 10.201 gazebo::physics::SphereShape Class Reference

Sphere collision shape.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::SphereShape:




### Public Member Functions

- **SphereShape** (**CollisionPtr** _parent)

  *Constructor.*
- virtual ~**SphereShape** ()

  *Destructor.*
- virtual void **FillMsg** (msgs::Geometry &_msg)

  *Fill in the values for a geomertry message.*
- double **GetRadius** () const

*Get the sphere's radius.*

- virtual void **Init** ()

  *Initialize the sphere.*

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  *Process a geometry message.*

- virtual void **SetRadius** (double _radius)

  *Set the size.*

- virtual void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale of the sphere.*

## Additional Inherited Members

### 10.201.1 Detailed Description

Sphere collision shape.

### 10.201.2 Constructor & Destructor Documentation

#### 10.201.2.1 gazebo::physics::SphereShape::SphereShape ( CollisionPtr *_parent* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent collision object. |

#### 10.201.2.2 virtual gazebo::physics::SphereShape::~SphereShape ( ) [virtual]

Destructor.

### 10.201.3 Member Function Documentation

#### 10.201.3.1 virtual void gazebo::physics::SphereShape::FillMsg ( msgs::Geometry & *_msg* ) [virtual]

Fill in the values for a geomertry message.

**Parameters**

| | | |
|---|---|---|
| out | *_msg* | The geometry message to fill. |

Implements **gazebo::physics::Shape** (p. 863).

#### 10.201.3.2 double gazebo::physics::SphereShape::GetRadius ( ) const

Get the sphere's radius.

**Returns**

Radius of the sphere.

#### 10.201.3.3 virtual void gazebo::physics::SphereShape::Init ( ) [virtual]

Initialize the sphere.

Implements **gazebo::physics::Shape** (p. 864).

#### 10.201.3.4 virtual void gazebo::physics::SphereShape::ProcessMsg ( const msgs::Geometry & *_msg* ) [virtual]

Process a geometry message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | The message to set values from. |

Implements **gazebo::physics::Shape** (p. 864).

#### 10.201.3.5 virtual void gazebo::physics::SphereShape::SetRadius ( double *_radius* ) [virtual]

Set the size.

**Parameters**

| | | |
|---|---|---|
| in | *_radius* | Radius of the sphere. |

Reimplemented in **gazebo::physics::SimbodySphereShape** (p. 944), and **gazebo::physics::DARTSphereShape** (p. 351).

Referenced by gazebo::physics::DARTSphereShape::SetRadius(), and gazebo::physics::SimbodySphereShape::SetRadius().

#### 10.201.3.6 virtual void gazebo::physics::SphereShape::SetScale ( const math::Vector3 & *_scale* ) [virtual]

Set the scale of the sphere.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | Scale to set the sphere to. |

Implements **gazebo::physics::Shape** (p. 864).

The documentation for this class was generated from the following file:

- **SphereShape.hh**

## 10.202 gazebo::common::SphericalCoordinates Class Reference

Convert spherical coordinates for planetary surfaces.

```
#include <commmon/common.hh>
```

## Public Types

- enum **SurfaceType** { **EARTH_WGS84** = 1 }

  *Unique identifiers for planetary surface models.*

## Public Member Functions

- **SphericalCoordinates** ()

  *Constructor.*
- **SphericalCoordinates** (const **SurfaceType** _type)

  *Constructor with surface type input.*
- **SphericalCoordinates** (const **SurfaceType** _type, const **math::Angle** &_latitude, const **math::Angle** &_longitude, double _elevation, const **math::Angle** &_heading)

  *Constructor with surface type, angle, and elevation inputs.*
- ∼**SphericalCoordinates** ()

  *Destructor.*
- double **GetElevationReference** () const

  *Get reference elevation in meters.*
- **math::Angle GetHeadingOffset** () const

  *Get heading offset for gazebo reference frame, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.*
- **math::Angle GetLatitudeReference** () const

  *Get reference geodetic latitude.*
- **math::Angle GetLongitudeReference** () const

  *Get reference longitude.*
- **SurfaceType GetSurfaceType** () const

  *Get SurfaceType currently in use.*
- **math::Vector3 GlobalFromLocal** (const **math::Vector3** &_xyz) const

  *Convert a Cartesian velocity vector in the local gazebo frame to a global Cartesian frame with components East, North, Up.*
- void **SetElevationReference** (double _elevation)

  *Set reference elevation above sea level in meters.*
- void **SetHeadingOffset** (const **math::Angle** &_angle)

  *Set heading angle offset for gazebo frame.*
- void **SetLatitudeReference** (const **math::Angle** &_angle)

  *Set reference geodetic latitude.*
- void **SetLongitudeReference** (const **math::Angle** &_angle)

  *Set reference longitude.*
- void **SetSurfaceType** (const **SurfaceType** &_type)

  *Set SurfaceType for planetary surface model.*
- **math::Vector3 SphericalFromLocal** (const **math::Vector3** &_xyz) const

  *Convert a Cartesian position vector to geodetic coordinates.*

### Static Public Member Functions

- static **SurfaceType Convert** (const std::string &_str)

  *Convert a string to a SurfaceType.*

## 10.202.1 Detailed Description

Convert spherical coordinates for planetary surfaces.

## 10.202.2 Member Enumeration Documentation

### 10.202.2.1 enum gazebo::common::SphericalCoordinates::SurfaceType

Unique identifiers for planetary surface models.

**Enumerator**

***EARTH_WGS84*** Model of reference ellipsoid for earth, based on WGS 84 standard. see wikipedia: World_-Geodetic_System

## 10.202.3 Constructor & Destructor Documentation

### 10.202.3.1 gazebo::common::SphericalCoordinates::SphericalCoordinates ( )

Constructor.

### 10.202.3.2 gazebo::common::SphericalCoordinates::SphericalCoordinates ( const SurfaceType *_type* )

Constructor with surface type input.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | SurfaceType specification. |

### 10.202.3.3 gazebo::common::SphericalCoordinates::SphericalCoordinates ( const SurfaceType *_type*, const math::Angle & *_latitude*, const math::Angle & *_longitude*, double *_elevation*, const math::Angle & *_heading* )

Constructor with surface type, angle, and elevation inputs.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | SurfaceType specification. |
| in | *_latitude* | Reference latitude. |
| in | *_longitude* | Reference longitude. |
| in | *_elevation* | Reference elevation. |
| in | *_heading* | Heading offset. |

#### 10.202.3.4 gazebo::common::SphericalCoordinates::~SphericalCoordinates ( )

Destructor.

### 10.202.4 Member Function Documentation

#### 10.202.4.1 static SurfaceType gazebo::common::SphericalCoordinates::Convert ( const std::string & _str ) [static]

Convert a string to a SurfaceType.

**Parameters**

| | | |
|---|---|---|
| in | *_str* | String to convert. |

**Returns**

Conversion to SurfaceType.

#### 10.202.4.2 double gazebo::common::SphericalCoordinates::GetElevationReference ( ) const

Get reference elevation in meters.

**Returns**

Reference elevation.

#### 10.202.4.3 math::Angle gazebo::common::SphericalCoordinates::GetHeadingOffset ( ) const

Get heading offset for gazebo reference frame, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.

**Returns**

Heading offset of gazebo reference frame.

#### 10.202.4.4 math::Angle gazebo::common::SphericalCoordinates::GetLatitudeReference ( ) const

Get reference geodetic latitude.

**Returns**

Reference geodetic latitude.

#### 10.202.4.5 math::Angle gazebo::common::SphericalCoordinates::GetLongitudeReference ( ) const

Get reference longitude.

**Returns**

Reference longitude.

#### 10.202.4.6 SurfaceType gazebo::common::SphericalCoordinates::GetSurfaceType ( ) const

Get SurfaceType currently in use.

**Returns**

Current SurfaceType value.

#### 10.202.4.7 math::Vector3 gazebo::common::SphericalCoordinates::GlobalFromLocal ( const math::Vector3 & _xyz ) const

Convert a Cartesian velocity vector in the local gazebo frame to a global Cartesian frame with components East, North, Up.

**Parameters**

| | | |
|---|---|---|
| in | *_xyz* | Cartesian vector in gazebo's world frame. |

**Returns**

Rotated vector with components (x,y,z): (East, North, Up).

#### 10.202.4.8 void gazebo::common::SphericalCoordinates::SetElevationReference ( double _elevation )

Set reference elevation above sea level in meters.

**Parameters**

| | | |
|---|---|---|
| in | *_elevation* | Reference elevation. |

#### 10.202.4.9 void gazebo::common::SphericalCoordinates::SetHeadingOffset ( const math::Angle & _angle )

Set heading angle offset for gazebo frame.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | Heading offset for gazebo frame. |

#### 10.202.4.10 void gazebo::common::SphericalCoordinates::SetLatitudeReference ( const math::Angle & _angle )

Set reference geodetic latitude.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | Reference geodetic latitude. |

#### 10.202.4.11 void gazebo::common::SphericalCoordinates::SetLongitudeReference ( const math::Angle & _angle )

Set reference longitude.

**Parameters**

| | | |
|---|---|---|
| in | *_angle* | Reference longitude. |

#### 10.202.4.12 void gazebo::common::SphericalCoordinates::SetSurfaceType ( const SurfaceType & *_type* )

Set SurfaceType for planetary surface model.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | SurfaceType value. |

#### 10.202.4.13 math::Vector3 gazebo::common::SphericalCoordinates::SphericalFromLocal ( const math::Vector3 & *_xyz* ) const

Convert a Cartesian position vector to geodetic coordinates.

**Parameters**

| | | |
|---|---|---|
| in | *_xyz* | Cartesian position vector in gazebo's world frame. |

**Returns**

Cooordinates: geodetic latitude (deg), longitude (deg), altitude above sea level (m).

The documentation for this class was generated from the following file:

- **SphericalCoordinates.hh**

## 10.203 gazebo::math::Spline Class Reference

Splines.

`#include <math/gzmath.hh>`

### Public Member Functions

- **Spline** ()
  *constructor*
- ~**Spline** ()
  *destructor*
- void **AddPoint** (const **Vector3** &_pt)
  *Adds a control point to the end of the spline.*
- void **Clear** ()
  *Clears all the points in the spline.*
- **Vector3 GetPoint** (unsigned int _index) const
  *Gets the detail of one of the control points of the spline.*
- unsigned int **GetPointCount** () const
  *Gets the number of control points in the spline.*

- **Vector3 GetTangent** (unsigned int _index) const

  *Get the tangent value for a point.*
- double **GetTension** () const

  *Get the tension value.*
- **Vector3 Interpolate** (double _t) const

  *Returns an interpolated point based on a parametric value over the whole series.*
- **Vector3 Interpolate** (unsigned int _fromIndex, double _t) const

  *Interpolates a single segment of the spline given a parametric value.*
- void **RecalcTangents** ()

  *Recalculates the tangents associated with this spline.*
- void **SetAutoCalculate** (bool _autoCalc)

  *Tells the spline whether it should automatically calculate tangents on demand as points are added.*
- void **SetTension** (double _t)

  *Set the tension parameter.*
- void **UpdatePoint** (unsigned int _index, const **Vector3** &_value)

  *Updates a single point in the spline.*

### Protected Attributes

- bool **autoCalc**

  *when true, the tangents are recalculated when the control point change*
- **Matrix4 coeffs**

  *Matrix of coefficients.*
- std::vector< **Vector3** > **points**

  *control points*
- std::vector< **Vector3** > **tangents**

  *tangents*
- double **tension**

  *Tension of 0 = Catmull-Rom spline, otherwise a Cardinal spline.*

## 10.203.1 Detailed Description

Splines.

## 10.203.2 Constructor & Destructor Documentation

#### 10.203.2.1 gazebo::math::Spline::Spline ( )

constructor

#### 10.203.2.2 gazebo::math::Spline::~Spline ( )

destructor

### 10.203.3 Member Function Documentation

#### 10.203.3.1 void gazebo::math::Spline::AddPoint ( const Vector3 & _pt )

Adds a control point to the end of the spline.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | point to add |

#### 10.203.3.2 void gazebo::math::Spline::Clear ( )

Clears all the points in the spline.

#### 10.203.3.3 Vector3 gazebo::math::Spline::GetPoint ( unsigned int _index ) const

Gets the detail of one of the control points of the spline.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | the control point index |

**Returns**

the control point, or [0,0,0] and a message on the error stream

#### 10.203.3.4 unsigned int gazebo::math::Spline::GetPointCount ( ) const

Gets the number of control points in the spline.

**Returns**

the count

#### 10.203.3.5 Vector3 gazebo::math::Spline::GetTangent ( unsigned int _index ) const

Get the tangent value for a point.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | the control point index |

#### 10.203.3.6 double gazebo::math::Spline::GetTension ( ) const

Get the tension value.

**Returns**

The value of the tension, which is between 0.0 and 1.0

#### 10.203.3.7 Vector3 gazebo::math::Spline::Interpolate ( double _t ) const

Returns an interpolated point based on a parametric value over the whole series.

**Parameters**

| | | |
|---|---|---|
| in | *_t* | parameter (range 0 to 1) |

#### 10.203.3.8 Vector3 gazebo::math::Spline::Interpolate ( unsigned int *_fromIndex*, double _t ) const

Interpolates a single segment of the spline given a parametric value.

**Parameters**

| | | |
|---|---|---|
| in | *_fromIndex* | The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1 |
| in | *_t* | Parametric value |

#### 10.203.3.9 void gazebo::math::Spline::RecalcTangents ( )

Recalculates the tangents associated with this spline.

**Remarks**

If you tell the spline not to update on demand by calling setAutoCalculate(false) then you must call this after completing your updates to the spline points.

#### 10.203.3.10 void gazebo::math::Spline::SetAutoCalculate ( bool *_autoCalc* )

Tells the spline whether it should automatically calculate tangents on demand as points are added.

**Remarks**

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the recalcTangents method when you are done.

**Parameters**

| | | |
|---|---|---|
| in | *_autoCalc* | If true, tangents are calculated for you whenever a point changes. If false, you must call reclacTangents to recalculate them when it best suits. |

#### 10.203.3.11 void gazebo::math::Spline::SetTension ( double _t )

Set the tension parameter.

A value of 0 = Catmull-Rom spline.

**Parameters**

| | | |
|---|---|---|
| in | *_t* | Tension value between 0.0 and 1.0 |

#### 10.203.3.12 void gazebo::math::Spline::UpdatePoint ( unsigned int _index, const Vector3 & _value )

Updates a single point in the spline.

**Remarks**

an error to the error stream is printed when the index is out of bounds

**Parameters**

| | | |
|---|---|---|
| in | *_index* | the control point index |
| in | *_value* | the new position |

### 10.203.4 Member Data Documentation

#### 10.203.4.1 bool gazebo::math::Spline::autoCalc [protected]

when true, the tangents are recalculated when the control point change

#### 10.203.4.2 Matrix4 gazebo::math::Spline::coeffs [protected]

Matrix of coefficients.

#### 10.203.4.3 std::vector<Vector3> gazebo::math::Spline::points [protected]

control points

#### 10.203.4.4 std::vector<Vector3> gazebo::math::Spline::tangents [protected]

tangents

#### 10.203.4.5 double gazebo::math::Spline::tension [protected]

Tension of 0 = Catmull-Rom spline, otherwise a Cardinal spline.

The documentation for this class was generated from the following file:

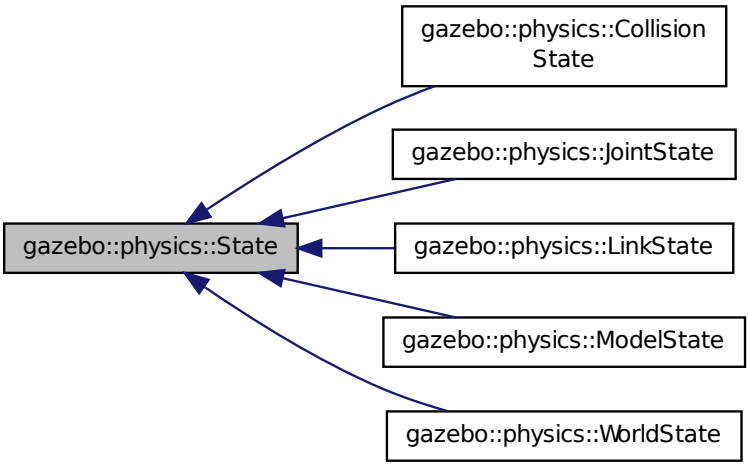
- **Spline.hh**

## 10.204 gazebo::physics::State Class Reference

**State** (p. 998) of an entity.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::State:




### Public Member Functions

- **State** ()
  *Default constructor.*
- **State** (const std::string &_name, const **common::Time** &_realTime, const **common::Time** &_simTime)
  *Constructor.*
- virtual ~**State** ()
  *Destructor.*
- std::string **GetName** () const
  *Get the name associated with this **State** (p. 998).*
- **common::Time GetRealTime** () const
  *Get the real time when this state was generated.*
- **common::Time GetSimTime** () const
  *Get the sim time when this state was generated.*
- **common::Time GetWallTime** () const
  *Get the wall time when this state was generated.*
- virtual void **Load** (const sdf::ElementPtr _elem)
  *Load state from SDF element.*
- **State operator-** (const **State** &_state) const
  *Subtraction operator.*
- **State** & **operator=** (const **State** &_state)

*Assignment operator.*

- void **SetName** (const std::string &_name)

  *Set the name associated with this **State** (p. 998).*
- virtual void **SetRealTime** (const **common::Time** &_time)

  *Set the real time when this state was generated.*
- virtual void **SetSimTime** (const **common::Time** &_time)

  *Set the sim time when this state was generated.*
- virtual void **SetWallTime** (const **common::Time** &_time)

  *Set the wall time when this state was generated.*

### Protected Attributes

- std::string **name**

  *Name associated with this **State** (p. 998).*
- **common::Time realTime**
- **common::Time simTime**
- **common::Time wallTime**

  *Times for the state data.*

## 10.204.1 Detailed Description

**State** (p. 998) of an entity.

This is the base class for all **State** (p. 998) information.

## 10.204.2 Constructor & Destructor Documentation

### 10.204.2.1 gazebo::physics::State::State ( )

Default constructor.

### 10.204.2.2 gazebo::physics::State::State ( const std::string & *_name*, const **common::Time** & *_realTime*, const **common::Time** & *_simTime* )

Constructor.

Construct a **State** (p. 998) object using some basic information.

**Parameters**

| | |
|---|---|
| *_name* | Name associated with the **State** (p. 998) information. This is typically the name of an **Entity** (p. 379). _realTime Clock time since simulation started. |
| *_simTime* | Simulation time associated with this **State** (p. 998) info. |

### 10.204.2.3 virtual gazebo::physics::State::~State ( ) [virtual]

Destructor.

### 10.204.3 Member Function Documentation

#### 10.204.3.1 std::string gazebo::physics::State::GetName ( ) const

Get the name associated with this **State** (p. 998).

**Returns**

Name associated with this state information. Typically a name of an **Entity** (p. 379).

#### 10.204.3.2 common::Time gazebo::physics::State::GetRealTime ( ) const

Get the real time when this state was generated.

**Returns**

Clock time since simulation was stated.

#### 10.204.3.3 common::Time gazebo::physics::State::GetSimTime ( ) const

Get the sim time when this state was generated.

**Returns**

Simulation time when the data was recorded.

#### 10.204.3.4 common::Time gazebo::physics::State::GetWallTime ( ) const

Get the wall time when this state was generated.

**Returns**

The absolute clock time when the **State** (p. 998) data was recorded.

#### 10.204.3.5 virtual void gazebo::physics::State::Load ( const sdf::ElementPtr *_elem* ) [virtual]

Load state from SDF element.

Populates the **State** (p. 998) information from data stored in an SDF::Element

**Parameters**

| | |
|---|---|
| *_elem* | Pointer to the SDF::Element |

Reimplemented in **gazebo::physics::ModelState** (p. 647), **gazebo::physics::LinkState** (p. 568), **gazebo::physics::WorldState** (p. 1175), **gazebo::physics::JointState** (p. 526), and **gazebo::physics::CollisionState** (p. 232).

#### 10.204.3.6 State gazebo::physics::State::operator- ( const State & _state ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to substract. |

**Returns**

The resulting state.

#### 10.204.3.7 State& gazebo::physics::State::operator= ( const State & _state )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **State** (p. 998) value |

**Returns**

this

#### 10.204.3.8 void gazebo::physics::State::SetName ( const std::string & _name )

Set the name associated with this **State** (p. 998).

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name associated with this state information. Typically the name of an **Entity** (p. 379). |

#### 10.204.3.9 virtual void gazebo::physics::State::SetRealTime ( const common::Time & _time ) [virtual]

Set the real time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Clock time since simulation was stated. |

Reimplemented in **gazebo::physics::ModelState** (p. 648), **gazebo::physics::LinkState** (p. 569), and **gazebo::physics::WorldState** (p. 1175).

#### 10.204.3.10 virtual void gazebo::physics::State::SetSimTime ( const common::Time & _time ) [virtual]

Set the sim time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Simulation time when the data was recorded. |

Reimplemented in **gazebo::physics::ModelState** (p. 649), **gazebo::physics::LinkState** (p. 569), and **gazebo::physics::WorldState** (p. 1176).

#### 10.204.3.11 virtual void gazebo::physics::State::SetWallTime ( const common::Time & *_time* ) [virtual]

Set the wall time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The absolute clock time when the **State** (p. 998) data was recorded. |

Reimplemented in **gazebo::physics::ModelState** (p. 649), **gazebo::physics::LinkState** (p. 570), and **gazebo::physics::WorldState** (p. 1176).

### 10.204.4 Member Data Documentation

#### 10.204.4.1 std::string gazebo::physics::State::name [protected]

Name associated with this **State** (p. 998).

#### 10.204.4.2 common::Time gazebo::physics::State::realTime [protected]

#### 10.204.4.3 common::Time gazebo::physics::State::simTime [protected]

#### 10.204.4.4 common::Time gazebo::physics::State::wallTime [protected]

Times for the state data.

The documentation for this class was generated from the following file:

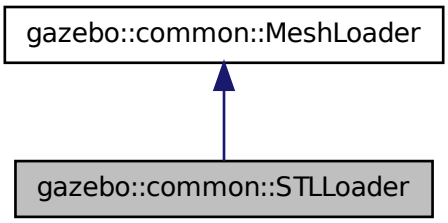
- **State.hh**

## 10.205 gazebo::common::STLLoader Class Reference

Class used to load STL mesh files.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::STLLoader:




### Public Member Functions

- **STLLoader** ()
  *Constructor.*
- virtual ~**STLLoader** ()
  *Destructor.*
- virtual **Mesh** * **Load** (const std::string &_filename)
  *Creates a new mesh and loads the data from a file.*

## 10.205.1 Detailed Description

Class used to load STL mesh files.

## 10.205.2 Constructor & Destructor Documentation

### 10.205.2.1 gazebo::common::STLLoader::STLLoader ( )

Constructor.

### 10.205.2.2 virtual gazebo::common::STLLoader::~STLLoader ( ) [virtual]

Destructor.

## 10.205.3 Member Function Documentation

### 10.205.3.1 virtual Mesh* gazebo::common::STLLoader::Load ( const std::string & *_filename* ) [virtual]

Creates a new mesh and loads the data from a file.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | the mesh file |

Implements **gazebo::common::MeshLoader** (p. 615).

The documentation for this class was generated from the following file:

- **STLLoader.hh**

## 10.206 gazebo::common::SubMesh Class Reference

A child mesh.

`#include <Mesh.hh>`

### Public Types

- enum **PrimitiveType** {
  **POINTS**, **LINES**, **LINESTRIPS**, **TRIANGLES**,
  **TRIFANS**, **TRISTRIPS** }

  *An enumeration of the geometric mesh primitives.*

### Public Member Functions

- **SubMesh** ()

  *Constructor.*
- **SubMesh** (const **SubMesh** *_mesh)

  *Copy Constructor.*
- virtual ~**SubMesh** ()

  *Destructor.*
- void **AddIndex** (unsigned int _i)

  *Add an index to the mesh.*
- void **AddNodeAssignment** (unsigned int _vertex, unsigned int _node, float _weight)

  *Add a vertex - skeleton node assignment.*
- void **AddNormal** (const **math::Vector3** &_n)

  *Add a normal to the mesh.*
- void **AddNormal** (double _x, double _y, double _z)

  *Add a normal to the mesh.*
- void **AddTexCoord** (double _u, double _v)

  *Add a texture coord to the mesh.*
- void **AddVertex** (const **math::Vector3** &_v)

  *Add a vertex to the mesh.*
- void **AddVertex** (double _x, double _y, double _z)

  *Add a vertex to the mesh.*
- void **Center** (const **math::Vector3** &_center=**math::Vector3::Zero**)

  *Move the center of the submesh to the given coordinate.*
- void **CopyNormals** (const std::vector< **math::Vector3** > &_norms)

  *Copy normals from a vector.*
- void **CopyVertices** (const std::vector< **math::Vector3** > &_verts)

  *Copy vertices from a vector.*
- void **FillArrays** (float **_vertArr, int **_indArr) const

*Put all the data into flat arrays.*

- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

  *Generate texture coordinates using spherical projection from center.*
- unsigned int **GetIndex** (unsigned int _i) const

  *Get an index.*
- unsigned int **GetIndexCount** () const

  *Return the number of indicies.*
- unsigned int **GetMaterialIndex** () const

  *Get the material index.*
- **math::Vector3 GetMax** () const

  *Get the maximun X, Y, Z values.*
- unsigned int **GetMaxIndex** () const

  *Get the highest index value.*
- **math::Vector3 GetMin** () const

  *Get the minimum X, Y, Z values.*
- std::string **GetName** () const

  *Get the name of this mesh.*
- **NodeAssignment GetNodeAssignment** (unsigned int _i) const

  *Get a vertex - skeleton node assignment.*
- unsigned int **GetNodeAssignmentsCount** () const

  *Return the number of vertex - skeleton node assignments.*
- **math::Vector3 GetNormal** (unsigned int _i) const

  *Get a normal.*
- unsigned int **GetNormalCount** () const

  *Return the number of normals.*
- **PrimitiveType GetPrimitiveType** () const

  *Get the primitive type.*
- **math::Vector2d GetTexCoord** (unsigned int _i) const

  *Get a tex coord.*
- unsigned int **GetTexCoordCount** () const

  *Return the number of texture coordinates.*
- **math::Vector3 GetVertex** (unsigned int _i) const

  *Get a vertex.*
- unsigned int **GetVertexCount** () const

  *Return the number of vertices.*
- unsigned int **GetVertexIndex** (const **math::Vector3** &_v) const

  *Get the index of the vertex.*
- bool **HasVertex** (const **math::Vector3** &_v) const

  *Return true if this submesh has the vertex.*
- void **RecalculateNormals** ()

  *Recalculate all the normals.*
- void **Scale** (double _factor)

  *Scale all vertices by _factor.*
- void **SetIndexCount** (unsigned int _count)

  *Resize the index array.*
- void **SetMaterialIndex** (unsigned int _index)

  *Set the material index.*

- void **SetName** (const std::string &_n)

  *Set the name of this mesh.*
- void **SetNormal** (unsigned int _i, const **math::Vector3** &_n)

  *Set a normal.*
- void **SetNormalCount** (unsigned int _count)

  *Resize the normal array.*
- void **SetPrimitiveType** (**PrimitiveType** _type)

  *Set the primitive type.*
- void **SetScale** (const **math::Vector3** &_factor)

  *Scale all vertices by the _factor vector.*
- void **SetSubMeshCenter** (**math::Vector3** _center)

  *Reset mesh center to geometric center.*
- void **SetTexCoord** (unsigned int _i, const **math::Vector2d** &_t)

  *Set a tex coord.*
- void **SetTexCoordCount** (unsigned int _count)

  *Resize the texture coordinate array.*
- void **SetVertex** (unsigned int _i, const **math::Vector3** &_v)

  *Set a vertex.*
- void **SetVertexCount** (unsigned int _count)

  *Resize the vertex array.*
- void **Translate** (const **math::Vector3** &_vec)

  *Move all vertices by _vec.*

### 10.206.1 Detailed Description

A child mesh.

### 10.206.2 Member Enumeration Documentation

#### 10.206.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

**Enumerator**

***POINTS***

***LINES***

***LINESTRIPS***

***TRIANGLES***

***TRIFANS***

***TRISTRIPS***

### 10.206.3 Constructor & Destructor Documentation

#### 10.206.3.1 gazebo::common::SubMesh::SubMesh ( )

Constructor.

#### 10.206.3.2 gazebo::common::SubMesh::SubMesh ( const SubMesh * _mesh )

Copy Constructor.

#### 10.206.3.3 virtual gazebo::common::SubMesh::~SubMesh ( ) [virtual]

Destructor.

### 10.206.4 Member Function Documentation

#### 10.206.4.1 void gazebo::common::SubMesh::AddIndex ( unsigned int _i )

Add an index to the mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | the new vertex index |

#### 10.206.4.2 void gazebo::common::SubMesh::AddNodeAssignment ( unsigned int _vertex, unsigned int _node, float _weight )

Add a vertex - skeleton node assignment.

**Parameters**

| | | |
|---|---|---|
| in | *_vertex* | the vertex index |
| in | *_node* | the node index |
| in | *_weight* | the weight (between 0 and 1) |

#### 10.206.4.3 void gazebo::common::SubMesh::AddNormal ( const math::Vector3 & _n )

Add a normal to the mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_n* | the normal |

#### 10.206.4.4 void gazebo::common::SubMesh::AddNormal ( double _x, double _y, double _z )

Add a normal to the mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | position along x |
| in | *_y* | position along y |
| in | *_z* | position along z |

#### 10.206.4.5 void gazebo::common::SubMesh::AddTexCoord ( double _u, double _v )

Add a texture coord to the mesh.

**Parameters**

| | | |
|---|---|---|
| in | _u | position along u |
| in | _v | position along v |

#### 10.206.4.6 void gazebo::common::SubMesh::AddVertex ( const math::Vector3 & _v )

Add a vertex to the mesh.

**Parameters**

| | | |
|---|---|---|
| in | _v | the new position |

#### 10.206.4.7 void gazebo::common::SubMesh::AddVertex ( double _x, double _y, double _z )

Add a vertex to the mesh.

**Parameters**

| | | |
|---|---|---|
| in | _x | position along x |
| in | _y | position along y |
| in | _z | position along z |

#### 10.206.4.8 void gazebo::common::SubMesh::Center ( const math::Vector3 & _center = math::Vector3::Zero )

Move the center of the submesh to the given coordinate.

This will move all the vertices.

**Parameters**

| | | |
|---|---|---|
| in | _center | Location of the mesh center. |

#### 10.206.4.9 void gazebo::common::SubMesh::CopyNormals ( const std::vector< math::Vector3 > & _norms )

Copy normals from a vector.

**Parameters**

| | | |
|---|---|---|
| in | _norms | to copy from |

#### 10.206.4.10 void gazebo::common::SubMesh::CopyVertices ( const std::vector< math::Vector3 > & _verts )

Copy vertices from a vector.

**Parameters**

| | | |
|---|---|---|
| in | *_verts* | the vertices to copy from |

#### 10.206.4.11 void gazebo::common::SubMesh::FillArrays ( float ** *_vertArr*, int ** *_indArr* ) const

Put all the data into flat arrays.

**Parameters**

| | | |
|---|---|---|
| in | *_verArr* | |
| in | *_indArr* | |

#### 10.206.4.12 void gazebo::common::SubMesh::GenSphericalTexCoord ( const math::Vector3 & *_center* )

Generate texture coordinates using spherical projection from center.

**Parameters**

| | | |
|---|---|---|
| in | *_center* | |

#### 10.206.4.13 unsigned int gazebo::common::SubMesh::GetIndex ( unsigned int *_i* ) const

Get an index.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | |

#### 10.206.4.14 unsigned int gazebo::common::SubMesh::GetIndexCount ( ) const

Return the number of indicies.

#### 10.206.4.15 unsigned int gazebo::common::SubMesh::GetMaterialIndex ( ) const

Get the material index.

#### 10.206.4.16 math::Vector3 gazebo::common::SubMesh::GetMax ( ) const

Get the maximun X, Y, Z values.

**Returns**

#### 10.206.4.17 unsigned int gazebo::common::SubMesh::GetMaxIndex ( ) const

Get the highest index value.

#### 10.206.4.18 math::Vector3 gazebo::common::SubMesh::GetMin ( ) const

Get the minimum X, Y, Z values.

**Returns**

#### 10.206.4.19 std::string gazebo::common::SubMesh::GetName ( ) const

Get the name of this mesh.

**Returns**

the name

#### 10.206.4.20 NodeAssignment gazebo::common::SubMesh::GetNodeAssignment ( unsigned int _i ) const

Get a vertex - skeleton node assignment.

**Parameters**

| | | |
|---|---|---|
| in | _i | the index of the assignment |

#### 10.206.4.21 unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount ( ) const

Return the number of vertex - skeleton node assignments.

#### 10.206.4.22 math::Vector3 gazebo::common::SubMesh::GetNormal ( unsigned int _i ) const

Get a normal.

**Parameters**

| | | |
|---|---|---|
| in | _i | the normal index |

**Returns**

the orientation of the normal, or throws an exception

#### 10.206.4.23 unsigned int gazebo::common::SubMesh::GetNormalCount ( ) const

Return the number of normals.

#### 10.206.4.24 PrimitiveType gazebo::common::SubMesh::GetPrimitiveType ( ) const

Get the primitive type.

**Returns**

the primitive type

#### 10.206.4.25 math::Vector2d gazebo::common::SubMesh::GetTexCoord ( unsigned int _i ) const

Get a tex coord.

**Parameters**

| | | |
|---|---|---|
| in | _i | the texture index |

**Returns**

the texture coordinates

#### 10.206.4.26 unsigned int gazebo::common::SubMesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

#### 10.206.4.27 math::Vector3 gazebo::common::SubMesh::GetVertex ( unsigned int _i ) const

Get a vertex.

**Parameters**

| | | |
|---|---|---|
| in | _i | the vertex index |

**Returns**

the position or throws an exception

#### 10.206.4.28 unsigned int gazebo::common::SubMesh::GetVertexCount ( ) const

Return the number of vertices.

#### 10.206.4.29 unsigned int gazebo::common::SubMesh::GetVertexIndex ( const math::Vector3 & _v ) const

Get the index of the vertex.

**Parameters**

| | | |
|---|---|---|
| in | _v | |

#### 10.206.4.30 bool gazebo::common::SubMesh::HasVertex ( const math::Vector3 & _v ) const

Return true if this submesh has the vertex.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | |

#### 10.206.4.31 void gazebo::common::SubMesh::RecalculateNormals ( )

Recalculate all the normals.

#### 10.206.4.32 void gazebo::common::SubMesh::Scale ( double *_factor* )

Scale all vertices by _factor.

**Parameters**

| | | |
|---|---|---|
| in | *_factor* | Scaling factor |

#### 10.206.4.33 void gazebo::common::SubMesh::SetIndexCount ( unsigned int *_count* )

Resize the index array.

**Parameters**

| | | |
|---|---|---|
| in | *_count* | the new size of the array |

#### 10.206.4.34 void gazebo::common::SubMesh::SetMaterialIndex ( unsigned int *_index* )

Set the material index.

Relates to the parent mesh material list

**Parameters**

| | | |
|---|---|---|
| in | *_index* | |

#### 10.206.4.35 void gazebo::common::SubMesh::SetName ( const std::string & *_n* )

Set the name of this mesh.

**Parameters**

| | | |
|---|---|---|
| in | *_n* | the name to set |

#### 10.206.4.36 void gazebo::common::SubMesh::SetNormal ( unsigned int *_i*, const math::Vector3 & *_n* )

Set a normal.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | the normal index |
| in | *_n* | the normal direction |

#### 10.206.4.37 void gazebo::common::SubMesh::SetNormalCount ( unsigned int *_count* )

Resize the normal array.

**Parameters**

| | | |
|---|---|---|
| in | *_count* | the new size of the array |

#### 10.206.4.38 void gazebo::common::SubMesh::SetPrimitiveType ( PrimitiveType *_type* )

Set the primitive type.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | the type |

#### 10.206.4.39 void gazebo::common::SubMesh::SetScale ( const math::Vector3 & *_factor* )

Scale all vertices by the _factor vector.

**Parameters**

| | | |
|---|---|---|
| in | *_factor* | Scaling vector |

#### 10.206.4.40 void gazebo::common::SubMesh::SetSubMeshCenter ( math::Vector3 *_center* )

Reset mesh center to geometric center.

**Parameters**

| | | |
|---|---|---|
| in | *_center* | |

#### 10.206.4.41 void gazebo::common::SubMesh::SetTexCoord ( unsigned int *_i,* const math::Vector2d & *_t* )

Set a tex coord.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | |
| in | *_t* | |

#### 10.206.4.42 void gazebo::common::SubMesh::SetTexCoordCount ( unsigned int _count )

Resize the texture coordinate array.

**Parameters**

| | | |
|---|---|---|
| in | *_count* | |

#### 10.206.4.43 void gazebo::common::SubMesh::SetVertex ( unsigned int _i, const math::Vector3 & _v )

Set a vertex.

**Parameters**

| | | |
|---|---|---|
| in | *_i* | the index |
| in | *_v* | the position |

#### 10.206.4.44 void gazebo::common::SubMesh::SetVertexCount ( unsigned int _count )

Resize the vertex array.

**Parameters**

| | | |
|---|---|---|
| in | *_count* | the new size of the array |

#### 10.206.4.45 void gazebo::common::SubMesh::Translate ( const math::Vector3 & _vec )

Move all vertices by _vec.

**Parameters**

| | | |
|---|---|---|
| in | *_vec* | Amount to translate vertices. |

The documentation for this class was generated from the following file:

- **Mesh.hh**

## 10.207 gazebo::transport::SubscribeOptions Class Reference

Options for a subscription.

`#include <transport/transport.hh>`

### Public Member Functions

- **SubscribeOptions** ()

  *Constructor.*
- bool **GetLatching** () const

*Are we latching?*

- std::string **GetMsgType** () const

  *Get the type of the topic we're subscribed to.*
- **NodePtr GetNode** () const

  *Get the node we're subscribed to.*
- std::string **GetTopic** () const

  *Get the topic we're subscribed to.*
- template<class M >
  void **Init** (const std::string &_topic, **NodePtr** _node, bool _latching)

  *Initialize the options.*
- void **Init** (const std::string &_topic, **NodePtr** _node, bool _latching)

  *Initialize the options.*

### 10.207.1 Detailed Description

Options for a subscription.

### 10.207.2 Constructor & Destructor Documentation

#### 10.207.2.1 gazebo::transport::SubscribeOptions::SubscribeOptions ( ) [inline]

Constructor.

### 10.207.3 Member Function Documentation

#### 10.207.3.1 bool gazebo::transport::SubscribeOptions::GetLatching ( ) const [inline]

Are we latching?

**Returns**

true if we're latching the latest message, false otherwise

#### 10.207.3.2 std::string gazebo::transport::SubscribeOptions::GetMsgType ( ) const [inline]

Get the type of the topic we're subscribed to.

**Returns**

The type of the topic we're subscribed to

#### 10.207.3.3 NodePtr gazebo::transport::SubscribeOptions::GetNode ( ) const [inline]

Get the node we're subscribed to.

**Returns**

The associated node

#### 10.207.3.4 std::string gazebo::transport::SubscribeOptions::GetTopic ( ) const [inline]

Get the topic we're subscribed to.

**Returns**

The topic we're subscribed to

#### 10.207.3.5 template<class M > void gazebo::transport::SubscribeOptions::Init ( const std::string & *_topic,* NodePtr *_node,* bool *_latching* ) [inline]

Initialize the options.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Topic we're subscribing to |
| in,out | *_node* | The associated node |
| in | *_latching* | If true, latch the latest message; if false, don't latch |

References gzthrow, and NULL.

Referenced by gazebo::transport::Node::Subscribe().

#### 10.207.3.6 void gazebo::transport::SubscribeOptions::Init ( const std::string & *_topic,* NodePtr *_node,* bool *_latching* ) [inline]

Initialize the options.

This version of init is only used when creating subscribers of raw data.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Topic we're subscribing to |
| in,out | *_node* | The associated node |
| in | *_latching* | If true, latch the latest message; if false, don't latch |

The documentation for this class was generated from the following file:

- **SubscribeOptions.hh**

## 10.208 gazebo::transport::Subscriber Class Reference

A subscriber to a topic.

`#include <transport/transport.hh>`

### Public Member Functions

- **Subscriber** (const std::string &_topic, **NodePtr** _node)

  *Constructor.*
- virtual ~**Subscriber** ()

*Destructor.*

- unsigned int **GetCallbackId** () const
- std::string **GetTopic** () const

  *Get the topic name.*
- void **SetCallbackId** (unsigned int _id)
- void **Unsubscribe** () const

  *Unsubscribe from the topic.*

### 10.208.1 Detailed Description

A subscriber to a topic.

### 10.208.2 Constructor & Destructor Documentation

#### 10.208.2.1 gazebo::transport::Subscriber::Subscriber ( const std::string & *_topic,* NodePtr *_node* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic we're subscribing to |
| in | *_node* | The associated node |

#### 10.208.2.2 virtual gazebo::transport::Subscriber::~Subscriber ( ) [virtual]

Destructor.

### 10.208.3 Member Function Documentation

#### 10.208.3.1 unsigned int gazebo::transport::Subscriber::GetCallbackId ( ) const

#### 10.208.3.2 std::string gazebo::transport::Subscriber::GetTopic ( ) const

Get the topic name.

**Returns**

The topic name

#### 10.208.3.3 void gazebo::transport::Subscriber::SetCallbackId ( unsigned int *_id* )

#### 10.208.3.4 void gazebo::transport::Subscriber::Unsubscribe ( ) const

Unsubscribe from the topic.

The documentation for this class was generated from the following file:

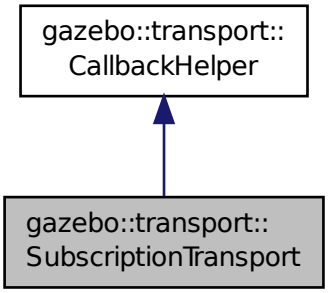
- **Subscriber.hh**

## 10.209 gazebo::transport::SubscriptionTransport Class Reference

transport/transport.hh

`#include <SubscriptionTransport.hh>`

Inheritance diagram for gazebo::transport::SubscriptionTransport:




### Public Member Functions

- **SubscriptionTransport** ()

  *Constructor.*
- virtual ~**SubscriptionTransport** ()

  *Destructor.*
- const **ConnectionPtr** & **GetConnection** () const

  *Get the connection we're using.*
- virtual bool **HandleData** (const std::string &_newdata, boost::function< void(uint32_t)> _cb, uint32_t _id)

  *Output a message to a connection.*
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  *Process new incoming message.*
- void **Init** (**ConnectionPtr** _conn, bool _latching)

  *Initialize the publication link.*
- virtual bool **IsLocal** () const

  *Is the callback local?*

### Additional Inherited Members

### 10.209.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers

## 10.209.2 Constructor & Destructor Documentation

### 10.209.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ( )

Constructor.

### 10.209.2.2 virtual gazebo::transport::SubscriptionTransport::~SubscriptionTransport ( ) [virtual]

Destructor.

## 10.209.3 Member Function Documentation

### 10.209.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection ( ) const

Get the connection we're using.

**Returns**

Pointer to the connection we're using

### 10.209.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData ( const std::string & *_newdata,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id* ) [virtual]

Output a message to a connection.

**Parameters**

| | | |
|---|---|---|
| in | *_newdata* | The message to be handled |

**Returns**

true if the message was handled successfully, false otherwise

**Parameters**

| | | |
|---|---|---|
| in | *_cb* | If non-null, callback to be invoked after transmission is complete. |
| in | *_id* | ID associated with the message data. |

Implements **gazebo::transport::CallbackHelper** (p. 182).

### 10.209.3.3 virtual bool gazebo::transport::SubscriptionTransport::HandleMessage ( MessagePtr *_newMsg* ) [virtual]

Process new incoming message.

**Parameters**

| | | |
|---|---|---|
| in | *_newMsg* | Incoming message to be processed |

**Returns**

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 183).

#### 10.209.3.4 void gazebo::transport::SubscriptionTransport::Init ( ConnectionPtr *_conn,* bool *_latching* )

Initialize the publication link.

**Parameters**

| | | |
|---|---|---|
| in | *_conn* | The connection to use |
| in | *_latching* | If true, latch the latest message; if false, don't latch |

#### 10.209.3.5 virtual bool gazebo::transport::SubscriptionTransport::IsLocal ( ) const [virtual]

Is the callback local?

**Returns**

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 183).

The documentation for this class was generated from the following file:

- **SubscriptionTransport.hh**

## 10.210 gazebo::physics::SurfaceParams Class Reference

**SurfaceParams** (p. 1020) defines various Surface contact parameters.

`#include <physics/physics.hh>`

### Public Member Functions

- **SurfaceParams** ()

  *Constructor.*
- virtual ~**SurfaceParams** ()

  *Destructor.*
- void **FillMsg** (msgs::Surface &_msg)

  *Fill in a surface message.*
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load the contact params.*
- virtual void **ProcessMsg** (const msgs::Surface &_msg)

### Public Attributes

- double **bounce**

  *bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.*
- double **bounceThreshold**

  *minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.*
- double **cfm**

  *Constraint Force Mixing parameter.*
- bool **collideWithoutContact**

  *Allow collision checking without generating a contact joint.*
- unsigned int **collideWithoutContactBitmask**

  *Custom collision filtering used when collideWithoutContact is true.*
- double **erp**

  *Error Reduction Parameter.*
- **math::Vector3 fdir1**

  *Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 1024)) of the friction pyramid.*
- double **kd**

  *spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 1022) and **SurfaceParams::erp** (p. 1023).*
- double **kp**

  *spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 1022) and **SurfaceParams::erp** (p. 1023).*
- double **maxVel**

  *Maximum interpenetration error correction velocity.*
- double **minDepth**

  *Minimum depth before ERP takes effect.*
- double **mu1**

  *Dry friction coefficient in the primary friction direction as defined by the friction pyramid.*
- double **mu2**

  *Dry friction coefficient in the second friction direction as defined by the friction pyramid.*
- double **slip1**

  *Artificial contact slip in the primary friction direction.*
- double **slip2**

  *Artificial contact slip in the secondary friction dirction.*

## 10.210.1 Detailed Description

**SurfaceParams** (p. 1020) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 260) constraint.

## 10.210.2 Constructor & Destructor Documentation

### 10.210.2.1 gazebo::physics::SurfaceParams::SurfaceParams ( )

Constructor.

#### 10.210.2.2 virtual gazebo::physics::SurfaceParams::~SurfaceParams ( ) [virtual]

Destructor.

### 10.210.3 Member Function Documentation

#### 10.210.3.1 void gazebo::physics::SurfaceParams::FillMsg ( msgs::Surface & *_msg* )

Fill in a surface message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Message to fill with this object's values. |

#### 10.210.3.2 virtual void gazebo::physics::SurfaceParams::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the contact params.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

#### 10.210.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg ( const msgs::Surface & *_msg* ) [virtual]

### 10.210.4 Member Data Documentation

#### 10.210.4.1 double gazebo::physics::SurfaceParams::bounce

bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.

**See Also**

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

#### 10.210.4.2 double gazebo::physics::SurfaceParams::bounceThreshold

minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.

**See Also**

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

#### 10.210.4.3 double gazebo::physics::SurfaceParams::cfm

Constraint Force Mixing parameter.

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_0 for more details.

#### 10.210.4.4 bool gazebo::physics::SurfaceParams::collideWithoutContact

Allow collision checking without generating a contact joint.

#### 10.210.4.5 unsigned int gazebo::physics::SurfaceParams::collideWithoutContactBitmask

Custom collision filtering used when collideWithoutContact is true.

#### 10.210.4.6 double gazebo::physics::SurfaceParams::erp

Error Reduction Parameter.

**See Also**

See for example `http://www.ode.org/ode-latest-userguide.html#sec_3_8_0` for more details.

#### 10.210.4.7 math::Vector3 gazebo::physics::SurfaceParams::fdir1

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 1024)) of the friction pyramid.

```
If undefined, a vector consstrained to be perpendicular
to the contact normal in the global y-z plane is used.
```

**See Also**

`http://www.ode.org/ode-latest-userguide.html#sec_7_3_7`

#### 10.210.4.8 double gazebo::physics::SurfaceParams::kd

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 1022) and **SurfaceParams::erp** (p. 1023).

**See Also**

See for example `http://www.ode.org/ode-latest-userguide.html#sec_3_8_2` for more details.

#### 10.210.4.9 double gazebo::physics::SurfaceParams::kp

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 1022) and **SurfaceParams::erp** (p. 1023).

**See Also**

See for example `http://www.ode.org/ode-latest-userguide.html#sec_3_8_2` for more details.

#### 10.210.4.10 double gazebo::physics::SurfaceParams::maxVel

Maximum interpenetration error correction velocity.

If set to 0, two objects interpenetrating each other will not be pushed apart.

**See Also**

See dWroldSetContactMaxCorrectingVel (http://www.ode.org/ode-latest-userguide.html#sec_5_2_0)

#### 10.210.4.11 double gazebo::physics::SurfaceParams::minDepth

Minimum depth before ERP takes effect.

**See Also**

See dWorldSetContactSurfaceLayer (http://www.ode.org/ode-latest-userguide.html#sec_5_2_0)

#### 10.210.4.12 double gazebo::physics::SurfaceParams::mu1

Dry friction coefficient in the primary friction direction as defined by the friction pyramid.

This is fdir1 if defined, otherwise, a vector consstrained to be perpendicular to the contact normal in the global y-z plane is used.

**See Also**

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

#### 10.210.4.13 double gazebo::physics::SurfaceParams::mu2

Dry friction coefficient in the second friction direction as defined by the friction pyramid.

This is fdir1 if defined, otherwise, a vector consstrained to be perpendicular to the contact normal in the global y-z plane is used.

**See Also**

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

#### 10.210.4.14 double gazebo::physics::SurfaceParams::slip1

Artificial contact slip in the primary friction direction.

**See Also**

See dContactSlip1 in http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

#### 10.210.4.15 double gazebo::physics::SurfaceParams::slip2

Artificial contact slip in the secondary friction dirction.

**See Also**

See dContactSlip2 in `http://www.ode.org/ode-latest-userguide.html#sec_7_3_7`

The documentation for this class was generated from the following file:

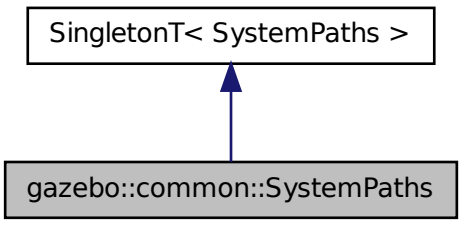
- **SurfaceParams.hh**

## 10.211 gazebo::common::SystemPaths Class Reference

Functions to handle getting system paths, keeps track of:

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::SystemPaths:




### Public Member Functions

- void **AddGazeboPaths** (const std::string &_path)

  *Add colon delimited paths to Gazebo install.*
- void **AddModelPaths** (const std::string &_path)

  *Add colon delimited paths to modelPaths.*
- void **AddOgrePaths** (const std::string &_path)

  *Add colon delimited paths to ogre install.*
- void **AddPluginPaths** (const std::string &_path)

  *Add colon delimited paths to plugins.*
- void **AddSearchPathSuffix** (const std::string &_suffix)

  *add _suffix to the list of path search suffixes*
- void **ClearGazeboPaths** ()

  *clear out SystemPaths::gazeboPaths*
- void **ClearModelPaths** ()

*clear out SystemPaths::modelPaths*

- void **ClearOgrePaths** ()

  *clear out SystemPaths::ogrePaths*

- void **ClearPluginPaths** ()

  *clear out SystemPaths::pluginPaths*

- std::string **FindFile** (const std::string &_filename, bool _searchLocalPath=true)

  *Find a file in the gazebo paths.*

- std::string **FindFileURI** (const std::string &_uri)

  *Find a file or path using a URI.*

- const std::list< std::string > & **GetGazeboPaths** ()

  *Get the gazebo install paths.*

- std::string **GetLogPath** () const

  *Get the log path.*

- const std::list< std::string > & **GetModelPaths** ()

  *Get the model paths.*

- const std::list< std::string > & **GetOgrePaths** ()

  *Get the ogre install paths.*

- const std::list< std::string > & **GetPluginPaths** ()

  *Get the plugin paths.*

- std::string **GetWorldPathExtension** ()

  *Returns the world path extension.*

## Public Attributes

- bool **gazeboPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1028)*

- bool **modelPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1028)*

- bool **ogrePathsFromEnv**

  *if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 1029)*

- bool **pluginPathsFromEnv**

  *if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 1029)*

## Additional Inherited Members

### 10.211.1 Detailed Description

Functions to handle getting system paths, keeps track of:

- SystemPaths::gazeboPaths - media paths containing worlds, models, sdf descriptions, material scripts, textures.
- SystemPaths::ogrePaths - ogre library paths. Should point to **Ogre** (p. 130) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

## 10.211.2 Member Function Documentation

### 10.211.2.1 void gazebo::common::SystemPaths::AddGazeboPaths ( const std::string & _path )

Add colon delimited paths to Gazebo install.

**Parameters**

| | | |
|---|---|---|
| in | *_path* | the directory to add |

### 10.211.2.2 void gazebo::common::SystemPaths::AddModelPaths ( const std::string & _path )

Add colon delimited paths to modelPaths.

**Parameters**

| | | |
|---|---|---|
| in | *_path* | the directory to add |

### 10.211.2.3 void gazebo::common::SystemPaths::AddOgrePaths ( const std::string & _path )

Add colon delimited paths to ogre install.

**Parameters**

| | | |
|---|---|---|
| in | *_path* | the directory to add |

### 10.211.2.4 void gazebo::common::SystemPaths::AddPluginPaths ( const std::string & _path )

Add colon delimited paths to plugins.

**Parameters**

| | | |
|---|---|---|
| in | *_path* | the directory to add |

### 10.211.2.5 void gazebo::common::SystemPaths::AddSearchPathSuffix ( const std::string & _suffix )

add _suffix to the list of path search suffixes

**Parameters**

| | | |
|---|---|---|
| in | *_suffix* | The suffix to add |

### 10.211.2.6 void gazebo::common::SystemPaths::ClearGazeboPaths ( )

clear out SystemPaths::gazeboPaths

#### 10.211.2.7 void gazebo::common::SystemPaths::ClearModelPaths ( )

clear out SystemPaths::modelPaths

#### 10.211.2.8 void gazebo::common::SystemPaths::ClearOgrePaths ( )

clear out SystemPaths::ogrePaths

#### 10.211.2.9 void gazebo::common::SystemPaths::ClearPluginPaths ( )

clear out SystemPaths::pluginPaths

#### 10.211.2.10 std::string gazebo::common::SystemPaths::FindFile ( const std::string & *_filename*, bool *_searchLocalPath* = `true` )

Find a file in the gazebo paths.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | Name of the file to find. |
| in | *_searchLocal-Path* | True to search in the current working directory. |

**Returns**

Returns full path name to file

#### 10.211.2.11 std::string gazebo::common::SystemPaths::FindFileURI ( const std::string & *_uri* )

Find a file or path using a URI.

**Parameters**

| | | |
|---|---|---|
| in | *_uri* | the uniform resource identifier |

**Returns**

Returns full path name to file

#### 10.211.2.12 const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ( )

Get the gazebo install paths.

**Returns**

a list of paths

#### 10.211.2.13 std::string gazebo::common::SystemPaths::GetLogPath ( ) const

Get the log path.

**Returns**

the path

#### 10.211.2.14 const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ( )

Get the model paths.

**Returns**

a list of paths

#### 10.211.2.15 const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ( )

Get the ogre install paths.

**Returns**

a list of paths

#### 10.211.2.16 const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ( )

Get the plugin paths.

**Returns**

a list of paths

#### 10.211.2.17 std::string gazebo::common::SystemPaths::GetWorldPathExtension ( )

Returns the world path extension.

**Returns**

Right now, it just returns "/worlds"

### 10.211.3 Member Data Documentation

#### 10.211.3.1 bool gazebo::common::SystemPaths::gazeboPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1028)

#### 10.211.3.2 bool gazebo::common::SystemPaths::modelPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1028)

#### 10.211.3.3 bool gazebo::common::SystemPaths::ogrePathsFromEnv

if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 1029)

#### 10.211.3.4 bool gazebo::common::SystemPaths::pluginPathsFromEnv

if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 1029)

The documentation for this class was generated from the following file:

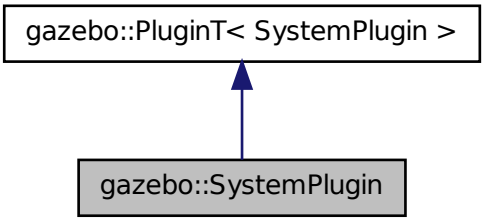
- **SystemPaths.hh**

## 10.212 gazebo::SystemPlugin Class Reference

A plugin loaded within the gzserver on startup.

`#include <Plugin.hh>`

Inheritance diagram for gazebo::SystemPlugin:




### Public Member Functions

- **SystemPlugin** ()
  *Constructor.*
- virtual ∼**SystemPlugin** ()
  *Destructor.*
- virtual void **Init** ()
  *Initialize the plugin.*
- virtual void **Load** (int _argc=0, char ∗∗_argv=**NULL**)=0
  *Load function.*
- virtual void **Reset** ()
  *Override this method for custom plugin reset behavior.*

### Additional Inherited Members

### 10.212.1 Detailed Description

A plugin loaded within the gzserver on startup.

See `reference`.

**Todo** how to make doxygen reference to the file gazebo.cc::g_plugins?

### 10.212.2 Constructor & Destructor Documentation

#### 10.212.2.1 gazebo::SystemPlugin::SystemPlugin ( ) [inline]

Constructor.

References gazebo::SYSTEM_PLUGIN, and gazebo::PluginT< SystemPlugin >::type.

#### 10.212.2.2 virtual gazebo::SystemPlugin::~SystemPlugin ( ) [inline], [virtual]

Destructor.

### 10.212.3 Member Function Documentation

#### 10.212.3.1 virtual void gazebo::SystemPlugin::Init ( ) [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

#### 10.212.3.2 virtual void gazebo::SystemPlugin::Load ( int *_argc* = 0, char ** *_argv* = NULL ) [pure virtual]

Load function.

Called before Gazebo is loaded. Must not block.

**Parameters**

| | |
|---|---|
| *_argc* | Number of command line arguments. |
| *_argv* | Array of command line arguments. |

#### 10.212.3.3 virtual void gazebo::SystemPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

- **Plugin.hh**

## 10.213 gazebo::common::Time Class Reference

A **Time** (p. 1031) class, can be used to hold wall- or sim-time.

`#include <common/common.hh>`

### Public Member Functions

- **Time** ()

*Constructors.*

- **Time** (const **Time** &_time)

  *Copy constructor.*
- **Time** (const struct timeval &_tv)

  *Constructor.*
- **Time** (const struct timespec &_tv)

  *Constructor.*
- **Time** (int32_t _sec, int32_t _nsec)

  *Constructor.*
- **Time** (double _time)

  *Constuctor.*
- virtual ∼**Time** ()

  *Destructor.*
- double **Double** () const

  *Get the time as a double.*
- float **Float** () const

  *Get the time as a float.*
- bool **operator!=** (const struct timeval &_tv) const

  *Equal to operator.*
- bool **operator!=** (const struct timespec &_tv) const

  *Equal to operator.*
- bool **operator!=** (const **Time** &_time) const

  *Equal to operator.*
- bool **operator!=** (double _time) const

  *Equal to operator.*
- **Time operator**∗ (const struct timeval &_tv) const

  *Multiplication operator.*
- **Time operator**∗ (const struct timespec &_tv) const

  *Multiplication operator.*
- **Time operator**∗ (const **Time** &_time) const

  *Multiplication operators.*
- const **Time** & **operator**∗**=** (const struct timeval &_tv)

  *Multiplication assignment operator.*
- const **Time** & **operator**∗**=** (const struct timespec &_tv)

  *Multiplication assignment operator.*
- const **Time** & **operator**∗**=** (const **Time** &_time)

  *Multiplication operators.*
- **Time operator+** (const struct timeval &_tv) const

  *Addition operators.*
- **Time operator+** (const struct timespec &_tv) const

  *Addition operators.*
- **Time operator+** (const **Time** &_time) const

  *Addition operators.*
- const **Time** & **operator+=** (const struct timeval &_tv)

  *Addition assignment operator.*
- const **Time** & **operator+=** (const struct timespec &_tv)

  *Addition assignment operator.*

- const **Time** & **operator+=** (const **Time** &_time)

  *Addition assignemtn operator.*
- **Time operator-** (const struct timeval &_tv) const

  *Subtraction operator.*
- **Time operator-** (const struct timespec &_tv) const

  *Subtraction operator.*
- **Time operator-** (const **Time** &_time) const

  *Subtraction operator.*
- const **Time** & **operator-=** (const struct timeval &_tv)

  *Subtraction assignment operator.*
- const **Time** & **operator-=** (const struct timespec &_tv)

  *Subtraction assignment operator.*
- const **Time** & **operator-=** (const **Time** &_time)

  *Subtraction assignment operator.*
- **Time operator**/ (const struct timeval &_tv) const

  *Division operator.*
- **Time operator**/ (const struct timespec &_tv) const

  *Division operator.*
- **Time operator**/ (const **Time** &_time) const

  *Division operator.*
- const **Time** & **operator**/**=** (const struct timeval &_tv)

  *Division assignment operator.*
- const **Time** & **operator**/**=** (const struct timespec &_tv)

  *Division assignment operator.*
- const **Time** & **operator**/**=** (const **Time** &time)

  *Division assignment operator.*
- bool **operator**< (const struct timeval &_tv) const

  *Less than operator.*
- bool **operator**< (const struct timespec &_tv) const

  *Less than operator.*
- bool **operator**< (const **Time** &_time) const

  *Less than operator.*
- bool **operator**< (double _time) const

  *Less than operator.*
- bool **operator**<**=** (const struct timeval &_tv) const

  *Less than or equal to operator.*
- bool **operator**<**=** (const struct timespec &_tv) const

  *Less than or equal to operator.*
- bool **operator**<**=** (const **Time** &_time) const

  *Less than or equal to operator.*
- bool **operator**<**=** (double _time) const

  *Less than or equal to operator.*
- **Time** & **operator=** (const struct timeval &_tv)

  *Assignment operator.*
- **Time** & **operator=** (const struct timespec &_tv)

  *Assignment operator.*
- **Time** & **operator=** (const **Time** &_time)

*Assignment operator.*

- bool **operator==** (const struct timeval &_tv) const

  *Equal to operator.*
- bool **operator==** (const struct timespec &_tv) const

  *Equal to operator.*
- bool **operator==** (const **Time** &_time) const

  *Equal to operator.*
- bool **operator==** (double _time) const

  *Equal to operator.*
- bool **operator**> (const struct timeval &_tv) const

  *Greater than operator.*
- bool **operator**> (const struct timespec &_tv) const

  *Greater than operator.*
- bool **operator**> (const **Time** &_time) const

  *Greater than operator.*
- bool **operator**> (double _time) const

  *Greater than operator.*
- bool **operator**>**=** (const struct timeval &_tv) const

  *Greater than or equal operator.*
- bool **operator**>**=** (const struct timespec &_tv) const

  *Greater than or equal operator.*
- bool **operator**>**=** (const **Time** &_time) const

  *Greater than or equal operator.*
- bool **operator**>**=** (double _time) const

  *Greater than or equal operator.*
- void **Set** (int32_t _sec, int32_t _nsec)

  *Set to sec and nsec.*
- void **Set** (double _seconds)

  *Set to seconds.*
- void **SetToWallTime** ()

  *Set the time to the wall time.*

## Static Public Member Functions

- static const **Time** & **GetWallTime** ()

  *Get the wall time.*
- static const std::string & **GetWallTimeAsISOString** ()

  *Get the wall time as an ISO string: YYYY-MM-DDTHH:MM:SS.*
- static double **MicToNano** (double _ms)

  *Convert microseconds to nanoseconds.*
- static double **MilToNano** (double _ms)

  *Convert milliseconds to nanoseconds.*
- static **Time MSleep** (unsigned int _ms)

  *Millisecond sleep.*
- static **Time NSleep** (unsigned int _ns)

  *Nano sleep.*
- static double **SecToNano** (double _sec)

*Convert seconds to nanoseconds.*

- static **Time Sleep** (const **common::Time** &_time)

  *Sleep for the specified time.*

### Public Attributes

- int32_t **nsec**

  *Nanoseconds.*

- int32_t **sec**

  *Seconds.*

### Static Public Attributes

- static const **Time Zero**

  *A static zero time variable set to common::Time(0, 0).*

### Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Time** &_time)

  *Stream insertion operator.*

- std::istream & **operator**>> (std::istream &_in, **gazebo::common::Time** &_time)

  *Stream extraction operator.*

## 10.213.1 Detailed Description

A **Time** (p. 1031) class, can be used to hold wall- or sim-time.

stored as sec and nano-sec.

## 10.213.2 Constructor & Destructor Documentation

### 10.213.2.1 gazebo::common::Time::Time ( )

Constructors.

### 10.213.2.2 gazebo::common::Time::Time ( const Time & *_time* )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | *time* | **Time** (p. 1031) to copy |

### 10.213.2.3 gazebo::common::Time::Time ( const struct timeval & *_tv* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | **Time** (p. 1031) to initialize to |

#### 10.213.2.4 gazebo::common::Time::Time ( const struct timespec & *_tv* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | **Time** (p. 1031) to initialize to |

#### 10.213.2.5 gazebo::common::Time::Time ( int32_t *_sec*, int32_t *_nsec* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_sec* | Seconds |
| in | *_nsec* | Nanoseconds |

#### 10.213.2.6 gazebo::common::Time::Time ( double *_time* )

Constuctor.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | **Time** (p. 1031) in double format sec.nsec |

#### 10.213.2.7 virtual gazebo::common::Time::~Time ( ) [virtual]

Destructor.

### 10.213.3 Member Function Documentation

#### 10.213.3.1 double gazebo::common::Time::Double ( ) const

Get the time as a double.

**Returns**

**Time** (p. 1031) as a double in seconds

#### 10.213.3.2 float gazebo::common::Time::Float ( ) const

Get the time as a float.

**Returns**

**Time** (p. 1031) as a float in seconds

#### 10.213.3.3 static const Time& gazebo::common::Time::GetWallTime ( ) [static]

Get the wall time.

**Returns**

the current time

Referenced by Joint_TEST::SpawnJoint().

#### 10.213.3.4 static const std::string& gazebo::common::Time::GetWallTimeAsISOString ( ) [static]

Get the wall time as an ISO string: YYYY-MM-DDTHH:MM:SS.

**Returns**

The current wall time as an ISO string.

#### 10.213.3.5 static double gazebo::common::Time::MicToNano ( double *_ms* ) [inline], [static]

Convert microseconds to nanoseconds.

**Parameters**

| | |
|---|---|
| *_ms* | microseconds |

**Returns**

nanoseconds

#### 10.213.3.6 static double gazebo::common::Time::MilToNano ( double *_ms* ) [inline], [static]

Convert milliseconds to nanoseconds.

**Parameters**

| | | |
|---|---|---|
| in | *_ms* | milliseconds |

**Returns**

nanoseconds

#### 10.213.3.7 static Time gazebo::common::Time::MSleep ( unsigned int *_ms* ) [static]

Millisecond sleep.

**Parameters**

| | | |
|---|---|---|
| in | *_ms* | milliseconds |

**Returns**

**Time** (p. 1031) actually slept

Referenced by Joint_TEST::SpawnJoint().

#### 10.213.3.8 static Time gazebo::common::Time::NSleep ( unsigned int _ns ) [static]

Nano sleep.

**Parameters**

| | | |
|---|---|---|
| in | *_ns* | nanoseconds |

**Returns**

**Time** (p. 1031) actually slept

#### 10.213.3.9 bool gazebo::common::Time::operator!= ( const struct timeval & _tv ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.10 bool gazebo::common::Time::operator!= ( const struct timespec & _tv ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.11 bool gazebo::common::Time::operator!= ( const Time & _time ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.12 bool gazebo::common::Time::operator!= ( double *_time* ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.13 Time gazebo::common::Time::operator* ( const struct timeval & *_tv* ) const

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | The scaling duration |

**Returns**

**Time** (p. 1031) instance

#### 10.213.3.14 Time gazebo::common::Time::operator* ( const struct timespec & *_tv* ) const

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the scaling duration |

**Returns**

**Time** (p. 1031) instance

#### 10.213.3.15 Time gazebo::common::Time::operator* ( const Time & *_time* ) const

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the scaling factor |

**Returns**

a scaled **Time** (p. 1031) instance

#### 10.213.3.16 const Time& gazebo::common::Time::operator*= ( const struct timeval & *_tv* )

Multiplication assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the scaling duration |

**Returns**

a reference to this instance

#### 10.213.3.17 const Time& gazebo::common::Time::operator*= ( const struct timespec & *_tv* )

Multiplication assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the scaling duration |

**Returns**

a reference to this instance

#### 10.213.3.18 const Time& gazebo::common::Time::operator*= ( const Time & *_time* )

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | scale factor |

**Returns**

a scaled **Time** (p. 1031) instance

#### 10.213.3.19 Time gazebo::common::Time::operator+ ( const struct timeval & *_tv* ) const

Addition operators.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to add |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.20 Time gazebo::common::Time::operator+ ( const struct timespec & _tv ) const

Addition operators.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to add |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.21 Time gazebo::common::Time::operator+ ( const Time & _time ) const

Addition operators.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The time to add |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.22 const Time& gazebo::common::Time::operator+= ( const struct timeval & _tv )

Addition assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to add |

**Returns**

a reference to this instance

#### 10.213.3.23 const Time& gazebo::common::Time::operator+= ( const struct timespec & _tv )

Addition assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to add |

**Returns**

a reference to this instance

#### 10.213.3.24 const Time& gazebo::common::Time::operator+= ( const Time & *_time* )

Addition assignemtn operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The time to add |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.25 Time gazebo::common::Time::operator- ( const struct timeval & *_tv* ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | The time to subtract |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.26 Time gazebo::common::Time::operator- ( const struct timespec & *_tv* ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | The time to subtract |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.27 Time gazebo::common::Time::operator- ( const Time & *_time* ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The time to subtract |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.28 const Time& gazebo::common::Time::operator-= ( const struct timeval & _tv )

Subtraction assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | The time to subtract |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.29 const Time& gazebo::common::Time::operator-= ( const struct timespec & _tv )

Subtraction assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | The time to subtract |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.30 const Time& gazebo::common::Time::operator-= ( const Time & _time )

Subtraction assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The time to subtract |

**Returns**

a reference to this instance

#### 10.213.3.31 Time gazebo::common::Time::operator/ ( const struct timeval & _tv ) const

Division operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | a timeval divisor |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.32 Time gazebo::common::Time::operator/ ( const struct timespec & _tv ) const

Division operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | a timespec divisor |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.33 Time gazebo::common::Time::operator/ ( const Time & _time ) const

Division operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the divisor |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.34 const Time& gazebo::common::Time::operator/= ( const struct timeval & _tv )

Division assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | a divisor |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.35 const Time& gazebo::common::Time::operator/= ( const struct timespec & _tv )

Division assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | a divisor |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.36 const Time& gazebo::common::Time::operator/= ( const Time & *time* )

Division assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *time* | the divisor |

**Returns**

a **Time** (p. 1031) instance

#### 10.213.3.37 bool gazebo::common::Time::operator< ( const struct timeval & *_tv* ) const

Less than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if tv is shorter than this, false otherwise

#### 10.213.3.38 bool gazebo::common::Time::operator< ( const struct timespec & *_tv* ) const

Less than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if tv is shorter than this, false otherwise

#### 10.213.3.39 bool gazebo::common::Time::operator< ( const Time & *_time* ) const

Less than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is shorter than this, false otherwise

#### 10.213.3.40 bool gazebo::common::Time::operator< ( double *_time* ) const

Less than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is shorter than this, false otherwise

#### 10.213.3.41 bool gazebo::common::Time::operator<= ( const struct timeval & *_tv* ) const

Less than or equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if tv is shorter than or equal to this, false otherwise

#### 10.213.3.42 bool gazebo::common::Time::operator<= ( const struct timespec & *_tv* ) const

Less than or equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if tv is shorter than or equal to this, false otherwise

#### 10.213.3.43 bool gazebo::common::Time::operator<= ( const Time & *_time* ) const

Less than or equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is shorter than or equal to this, false otherwise

#### 10.213.3.44 bool gazebo::common::Time::operator<= ( double *_time* ) const

Less than or equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is shorter than or equal to this, false otherwise

#### 10.213.3.45 Time& gazebo::common::Time::operator= ( const struct timeval & *_tv* )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the new time |

**Returns**

a reference to this instance

#### 10.213.3.46 Time& gazebo::common::Time::operator= ( const struct timespec & *_tv* )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the new time |

**Returns**

a reference to this instance

#### 10.213.3.47 Time& gazebo::common::Time::operator= ( const Time & *_time* )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the new time |

**Returns**

a reference to this instance

#### 10.213.3.48 bool gazebo::common::Time::operator== ( const struct timeval & *_tv* ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.49 bool gazebo::common::Time::operator== ( const struct timespec & *_tv* ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.50 bool gazebo::common::Time::operator== ( const Time & *_time* ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.51 bool gazebo::common::Time::operator== ( double *_time* ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare to |

**Returns**

true if values are the same, false otherwise

#### 10.213.3.52 bool gazebo::common::Time::operator> ( const struct timeval & *_tv* ) const

Greater than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if time is greater than this, false otherwise

#### 10.213.3.53 bool gazebo::common::Time::operator> ( const struct timespec & *_tv* ) const

Greater than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if time is greater than this, false otherwise

#### 10.213.3.54 bool gazebo::common::Time::operator> ( const Time & *_time* ) const

Greater than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is greater than this, false otherwise

#### 10.213.3.55 bool gazebo::common::Time::operator> ( double *_time* ) const

Greater than operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is greater than this, false otherwise

#### 10.213.3.56 bool gazebo::common::Time::operator>= ( const struct timeval & *_tv* ) const

Greater than or equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if tv is greater than or equal to this, false otherwise

#### 10.213.3.57 bool gazebo::common::Time::operator>= ( const struct timespec & *_tv* ) const

Greater than or equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_tv* | the time to compare with |

**Returns**

true if tv is greater than or equal to this, false otherwise

#### 10.213.3.58 bool gazebo::common::Time::operator>= ( const Time & *_time* ) const

Greater than or equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is greater than or equal to this, false otherwise

#### 10.213.3.59 bool gazebo::common::Time::operator>= ( double *_time* ) const

Greater than or equal operator.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | the time to compare with |

**Returns**

true if time is greater than or equal to this, false otherwise

#### 10.213.3.60 static double gazebo::common::Time::SecToNano ( double *_sec* ) [inline],[static]

Convert seconds to nanoseconds.

**Parameters**

| | | |
|---|---|---|
| in | *_sec* | duration in seconds |

**Returns**

nanoseconds

#### 10.213.3.61 void gazebo::common::Time::Set ( int32_t *_sec*, int32_t *_nsec* )

Set to sec and nsec.

**Parameters**

| | | |
|---|---|---|
| in | *_sec* | Seconds |
| in | *_nsec* | Nanoseconds |

#### 10.213.3.62 void gazebo::common::Time::Set ( double *_seconds* )

Set to seconds.

**Parameters**

| | | |
|---|---|---|
| in | *_seconds* | Number of seconds |

#### 10.213.3.63 void gazebo::common::Time::SetToWallTime ( )

Set the time to the wall time.

#### 10.213.3.64 static Time gazebo::common::Time::Sleep ( const common::Time & *_time* ) [static]

Sleep for the specified time.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Sleep time |

**Returns**

**Time** (p. 1031) actually slept

### 10.213.4 Friends And Related Function Documentation

#### 10.213.4.1 std::ostream& operator<< ( std::ostream & *_out,* const gazebo::common::Time & *_time* ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | the output stream |
| in | *_time* | time to write to the stream |

**Returns**

the output stream

#### 10.213.4.2 std::istream& operator>> ( std::istream & *_in,* gazebo::common::Time & *_time* ) [friend]

Stream extraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_in* | the input stream |
| in | *_time* | time to read from to the stream |

**Returns**

the input stream

### 10.213.5 Member Data Documentation

#### 10.213.5.1 int32_t gazebo::common::Time::nsec

Nanoseconds.

#### 10.213.5.2 int32_t gazebo::common::Time::sec

Seconds.

#### 10.213.5.3 const Time gazebo::common::Time::Zero [static]

A static zero time variable set to common::Time(0, 0).

Referenced by Joint_TEST::SpawnJoint().

The documentation for this class was generated from the following file:

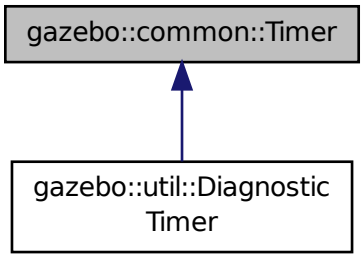
- **Time.hh**

## 10.214 gazebo::common::Timer Class Reference

A timer class, used to time things in real world walltime.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Timer:




### Public Member Functions

- **Timer** ()

  *Constructor.*
- virtual ~**Timer** ()

  *Destructor.*
- **Time GetElapsed** () const

  *Get the elapsed time.*
- bool **GetRunning** () const

  *Returns true if the timer is running.*
- virtual void **Start** ()

  *Start the timer.*
- virtual void **Stop** ()

  *Stop the timer.*

### Friends

- std::ostream & **operator**<< (std::ostream &out, const **gazebo::common::Timer** &t)

  *Stream operator friendly.*

### 10.214.1 Detailed Description

A timer class, used to time things in real world walltime.

### 10.214.2 Constructor & Destructor Documentation

#### 10.214.2.1 gazebo::common::Timer::Timer ( )

Constructor.

#### 10.214.2.2 virtual gazebo::common::Timer::~Timer ( ) [virtual]

Destructor.

### 10.214.3 Member Function Documentation

#### 10.214.3.1 Time gazebo::common::Timer::GetElapsed ( ) const

Get the elapsed time.

**Returns**

The time

#### 10.214.3.2 bool gazebo::common::Timer::GetRunning ( ) const

Returns true if the timer is running.

**Returns**

Tue if the timer has been started and not stopped.

#### 10.214.3.3 virtual void gazebo::common::Timer::Start ( ) [virtual]

Start the timer.

Reimplemented in **gazebo::util::DiagnosticTimer** (p. 369).

#### 10.214.3.4 virtual void gazebo::common::Timer::Stop ( ) [virtual]

Stop the timer.

Reimplemented in **gazebo::util::DiagnosticTimer** (p. 369).

### 10.214.4 Friends And Related Function Documentation

#### 10.214.4.1 std::ostream& operator<< ( std::ostream & *out,* const gazebo::common::Timer & *t* ) [friend]

Stream operator friendly.

The documentation for this class was generated from the following file:

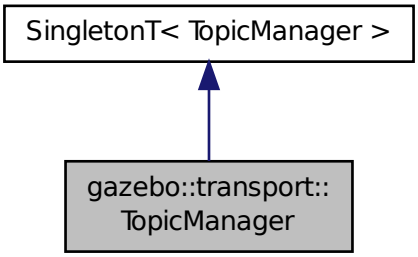
- **Timer.hh**

## 10.215 gazebo::transport::TopicManager Class Reference

Manages topics and their subscriptions.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::TopicManager:




### Public Types

- typedef std::map< std::string,
  std::list< **NodePtr** > > **SubNodeMap**

  *A map of string->list of **Node** (p. 672) pointers.*

### Public Member Functions

- void **AddNode** (**NodePtr** _node)

  *Add a node to the manager.*
- void **AddNodeToProcess** (**NodePtr** _ptr)

  *Add a node to the list of nodes that requires processing.*
- template<typename M >
  **PublisherPtr Advertise** (const std::string &_topic, unsigned int _queueLimit, double _hzRate)

  *Advertise on a topic.*
- void **ClearBuffers** ()

  *Clear all buffers.*
- void **ConnectPubToSub** (const std::string &_topic, const **SubscriptionTransportPtr** _sublink)

  ***Connection** (p. 247) a local **Publisher** (p. 757) to a remote **Subscriber** (p. 1016).*
- void **ConnectSubscribers** (const std::string &_topic)

  *Connect all subscribers on a topic to known publishers.*
- void **ConnectSubToPub** (const msgs::Publish &_pub)

  *Connect a local **Subscriber** (p. 1016) to a remote **Publisher** (p. 757).*
- void **DisconnectPubFromSub** (const std::string &_topic, const std::string &_host, unsigned int _port)

  *Disconnect a local publisher from a remote subscriber.*
- void **DisconnectSubFromPub** (const std::string &_topic, const std::string &_host, unsigned int _port)

*Disconnect all local subscribers from a remote publisher.*

- **PublicationPtr FindPublication** (const std::string &_topic)

  *Find a publication object by topic.*

- void **Fini** ()

  *Finalize the manager.*

- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  *Get all the topic namespaces.*

- void **Init** ()

  *Initialize the manager.*

- bool **IsAdvertised** (const std::string &_topic)

  *Has the topic been advertised?*

- void **PauseIncoming** (bool _pause)

  *Pause or unpause processing of incoming messages.*

- void **ProcessNodes** (bool _onlyOut=false)

  *Process all nodes under management.*

- void **Publish** (const std::string &_topic, **MessagePtr** _message, boost::function< void(uint32_t)> _cb, uint32_t _id)

  *Send a message.*

- void **RegisterTopicNamespace** (const std::string &_name)

  *Register a new topic namespace.*

- void **RemoveNode** (unsigned int _id)

  *Remove a node by its id.*

- **SubscriberPtr Subscribe** (const **SubscribeOptions** &_options)

  *Subscribe to a topic.*

- void **Unadvertise** (const std::string &_topic)

  *Unadvertise a topic.*

- void **Unsubscribe** (const std::string &_topic, const **NodePtr** &_sub)

  *Unsubscribe from a topic.*

- **PublicationPtr UpdatePublications** (const std::string &_topic, const std::string &_msgType)

  *Update our list of advertised topics.*

## Additional Inherited Members

### 10.215.1 Detailed Description

Manages topics and their subscriptions.

### 10.215.2 Member Typedef Documentation

#### 10.215.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

A map of string->list of **Node** (p. 672) pointers.

### 10.215.3 Member Function Documentation

#### 10.215.3.1 void gazebo::transport::TopicManager::AddNode ( NodePtr _node )

Add a node to the manager.

**Parameters**

| | | |
|---|---|---|
| in,out | *_node* | The node to be added |

#### 10.215.3.2 void gazebo::transport::TopicManager::AddNodeToProcess ( NodePtr _ptr )

Add a node to the list of nodes that requires processing.

**Parameters**

| | | |
|---|---|---|
| in | *_ptr* | **Node** (p. 672) to process. |

#### 10.215.3.3 template<typename M > PublisherPtr gazebo::transport::TopicManager::Advertise ( const std::string & _topic, unsigned int _queueLimit, double _hzRate ) [inline]

Advertise on a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The name of the topic |
| in | *_queueLimit* | The maximum number of outgoing messages to queue |
| in | *_hz* | Update rate for the publisher. Units are 1.0/seconds. |

**Returns**

Pointer to the newly created **Publisher** (p. 757)

References FindPublication(), GZ_ASSERT, gzthrow, SingletonT< T >::Instance(), NULL, and UpdatePublications().

#### 10.215.3.4 void gazebo::transport::TopicManager::ClearBuffers ( )

Clear all buffers.

#### 10.215.3.5 void gazebo::transport::TopicManager::ConnectPubToSub ( const std::string & _topic, const SubscriptionTransportPtr _sublink )

**Connection** (p. 247) a local **Publisher** (p. 757) to a remote **Subscriber** (p. 1016).

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to use |
| in | *_sublink* | The subscription transport object to use |

#### 10.215.3.6 void gazebo::transport::TopicManager::ConnectSubscribers ( const std::string & _topic )

Connect all subscribers on a topic to known publishers.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to be connected |

#### 10.215.3.7 void gazebo::transport::TopicManager::ConnectSubToPub ( const msgs::Publish & _pub )

Connect a local **Subscriber** (p. 1016) to a remote **Publisher** (p. 757).

**Parameters**

| | | |
|---|---|---|
| in | *_pub* | The publish object to use |

#### 10.215.3.8 void gazebo::transport::TopicManager::DisconnectPubFromSub ( const std::string & _topic, const std::string & _host, unsigned int _port )

Disconnect a local publisher from a remote subscriber.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to be disconnected |
| in | *_host* | The host to be disconnected |
| in | *_port* | The port to be disconnected |

#### 10.215.3.9 void gazebo::transport::TopicManager::DisconnectSubFromPub ( const std::string & _topic, const std::string & _host, unsigned int _port )

Disconnect all local subscribers from a remote publisher.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to be disconnected |
| in | *_host* | The host to be disconnected |
| in | *_port* | The port to be disconnected |

#### 10.215.3.10 PublicationPtr gazebo::transport::TopicManager::FindPublication ( const std::string & _topic )

Find a publication object by topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to search for |

**Returns**

Pointer to the publication object, if found (can be null)

Referenced by Advertise().

#### 10.215.3.11 void gazebo::transport::TopicManager::Fini ( )

Finalize the manager.

#### 10.215.3.12 void gazebo::transport::TopicManager::GetTopicNamespaces ( std::list< std::string > & *_namespaces* )

Get all the topic namespaces.

**Parameters**

| | | |
|---|---|---|
| out | *_namespaces* | The list of namespaces will be written here |

#### 10.215.3.13 void gazebo::transport::TopicManager::Init ( )

Initialize the manager.

#### 10.215.3.14 bool gazebo::transport::TopicManager::IsAdvertised ( const std::string & *_topic* )

Has the topic been advertised?

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The name of the topic to check |

**Returns**

true if the topic has been advertised, false otherwise

#### 10.215.3.15 void gazebo::transport::TopicManager::PauseIncoming ( bool *_pause* )

Pause or unpause processing of incoming messages.

**Parameters**

| | | |
|---|---|---|
| in | *_pause* | If true pause processing; otherwse unpause |

#### 10.215.3.16 void gazebo::transport::TopicManager::ProcessNodes ( bool *_onlyOut* = false )

Process all nodes under management.

**Parameters**

| | | |
|---|---|---|
| in | *_onlyOut* | True means only outbound messages on nodes will be sent. False means nodes process both outbound and inbound messages |

#### 10.215.3.17 void gazebo::transport::TopicManager::Publish ( const std::string & *_topic,* MessagePtr *_message,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id* )

Send a message.

Use a **Publisher** (p. 757) instead of calling this function directly.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | Name of the topic |
| in | *_message* | The message to send. |
| in | *_cb* | Callback, used when the publish is completed. |
| in | *_id* | ID associated with the message. |

#### 10.215.3.18 void gazebo::transport::TopicManager::RegisterTopicNamespace ( const std::string & *_name* )

Register a new topic namespace.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the new namespace |

#### 10.215.3.19 void gazebo::transport::TopicManager::RemoveNode ( unsigned int *_id* )

Remove a node by its id.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | The ID of the node to be removed |

#### 10.215.3.20 SubscriberPtr gazebo::transport::TopicManager::Subscribe ( const SubscribeOptions & *_options* )

Subscribe to a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_options* | The options to use for the subscription |

**Returns**

Pointer to the newly created subscriber

#### 10.215.3.21 void gazebo::transport::TopicManager::Unadvertise ( const std::string & *_topic* )

Unadvertise a topic.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to be unadvertised |

#### 10.215.3.22 void gazebo::transport::TopicManager::Unsubscribe ( const std::string & *_topic*, const NodePtr & *_sub* )

Unsubscribe from a topic.

Use a **Subscriber** (p. 1016) rather than calling this function directly

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to unsubscribe from |
| in | *_sub* | The node to unsubscribe |

#### 10.215.3.23 PublicationPtr gazebo::transport::TopicManager::UpdatePublications ( const std::string & *_topic*, const std::string & *_msgType* )

Update our list of advertised topics.

**Parameters**

| | | |
|---|---|---|
| in | *_topic* | The topic to be updated |
| in | *_msgType* | The type of the topic to be updated |

**Returns**

True if the provided params define a new publisher, false otherwise

Referenced by Advertise().

The documentation for this class was generated from the following file:

- **TopicManager.hh**

## 10.216 gazebo::physics::TrajectoryInfo Struct Reference

`#include <Actor.hh>`

### Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

### 10.216.1 Member Data Documentation

#### 10.216.1.1 double gazebo::physics::TrajectoryInfo::duration

#### 10.216.1.2 double gazebo::physics::TrajectoryInfo::endTime

#### 10.216.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

#### 10.216.1.4 double gazebo::physics::TrajectoryInfo::startTime

#### 10.216.1.5 bool gazebo::physics::TrajectoryInfo::translated

#### 10.216.1.6 std::string gazebo::physics::TrajectoryInfo::type

The documentation for this struct was generated from the following file:

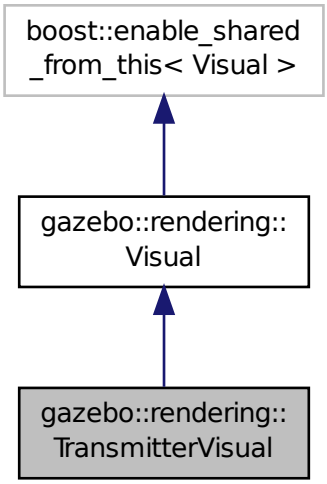
- **Actor.hh**

## 10.217 gazebo::rendering::TransmitterVisual Class Reference

Visualization for the wireless propagation data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::TransmitterVisual:




### Public Member Functions

- **TransmitterVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

*Constructor.*

- virtual ~**TransmitterVisual** ()

  *Destructor.*

- virtual void **Load** ()

  *Documentation inherited from parent.*

- virtual void **Update** ()

  *Function that runs on the OGRE thread to refresh the UI.*

## Additional Inherited Members

### 10.217.1 Detailed Description

Visualization for the wireless propagation data.

### 10.217.2 Constructor & Destructor Documentation

#### 10.217.2.1 gazebo::rendering::TransmitterVisual::TransmitterVisual ( const std::string & *_name,* VisualPtr *_vis,* const std::string & *_topicName* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual. |
| in | *_vis* | Pointer to the parent **Visual** (p. 1121). |
| in | *_topicName* | Name of the topic that has laser data. |

#### 10.217.2.2 virtual gazebo::rendering::TransmitterVisual::~TransmitterVisual ( ) [virtual]

Destructor.

### 10.217.3 Member Function Documentation

#### 10.217.3.1 virtual void gazebo::rendering::TransmitterVisual::Load ( ) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 1135).

#### 10.217.3.2 virtual void gazebo::rendering::TransmitterVisual::Update ( ) [virtual]

Function that runs on the OGRE thread to refresh the UI.

The documentation for this class was generated from the following file:

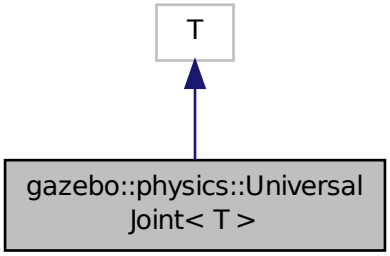
- **TransmitterVisual.hh**

## 10.218 gazebo::physics::UniversalJoint< T > Class Template Reference

A universal joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::UniversalJoint< T >:




### Public Member Functions

- **UniversalJoint** (**BasePtr** _parent)
  *Constructor.*
- virtual ~**UniversalJoint** ()
  *Destuctor.*
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
  *Load a **UniversalJoint** (p. 1064).*

### 10.218.1 Detailed Description

**template<class T>class gazebo::physics::UniversalJoint< T >**

A universal joint.

### 10.218.2 Constructor & Destructor Documentation

#### 10.218.2.1 template<class T> gazebo::physics::UniversalJoint< T >::UniversalJoint ( BasePtr *_parent* ) [inline],[explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_parent* | Parent link of the univeral joint. |

#### 10.218.2.2 template<class T> virtual gazebo::physics::UniversalJoint< T >::~UniversalJoint ( ) [inline], [virtual]

Destuctor.

### 10.218.3 Member Function Documentation

#### 10.218.3.1 template<class T> virtual unsigned int gazebo::physics::UniversalJoint< T >::GetAngleCount ( ) const [inline], [virtual]

#### 10.218.3.2 template<class T> virtual void gazebo::physics::UniversalJoint< T >::Load ( sdf::ElementPtr _sdf ) [inline], [virtual]

Load a **UniversalJoint** (p. 1064).

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF values to load from. |

Reimplemented in **gazebo::physics::SimbodyUniversalJoint** (p. 949), and **gazebo::physics::DARTUniversalJoint** (p. 356).

The documentation for this class was generated from the following file:

- **UniversalJoint.hh**

## 10.219 gazebo::common::UpdateInfo Class Reference

Information for use in an update event.

`#include <common/common.hh>`

### Public Attributes

- **common::Time realTime**
  *Current real time.*
- **common::Time simTime**
  *Current simulation time.*
- std::string **worldName**
  *Name of the world.*

### 10.219.1 Detailed Description

Information for use in an update event.

### 10.219.2 Member Data Documentation

#### 10.219.2.1 common::Time gazebo::common::UpdateInfo::realTime

Current real time.

#### 10.219.2.2 common::Time gazebo::common::UpdateInfo::simTime

Current simulation time.

#### 10.219.2.3 std::string gazebo::common::UpdateInfo::worldName

Name of the world.

The documentation for this class was generated from the following file:

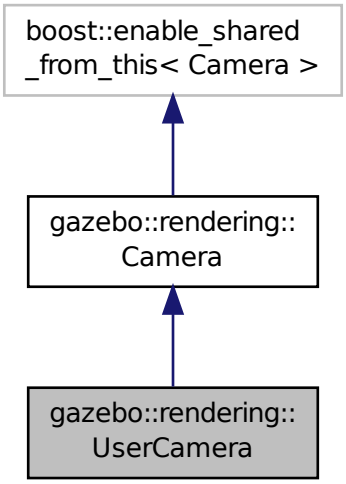
- **UpdateInfo.hh**

## 10.220 gazebo::rendering::UserCamera Class Reference

A camera used for user visualization of a scene.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::UserCamera:




### Public Member Functions

- **UserCamera** (const std::string &_name, **ScenePtr** _scene)

  *Constructor.*
- virtual ~**UserCamera** ()

  *Destructor.*
- void **EnableViewController** (bool _value) const

  *Set whether the view controller is enabled.*

- void **Fini** ()

  *Finialize.*
- float **GetAvgFPS** () const

  *Get the average frames per second.*
- **GUIOverlay** * **GetGUIOverlay** ()

  *Get the GUI overlay.*
- virtual unsigned int **GetImageHeight** () const

  *Get the height of the image.*
- virtual unsigned int **GetImageWidth** () const

  *Get the width of the image.*
- float **GetTriangleCount** () const

  *Get the triangle count.*
- std::string **GetViewControllerTypeString** ()

  *Get current view controller type.*
- **VisualPtr GetVisual** (const **math::Vector2i** &_mousePos, std::string &_mod)

  *Get an entity at a pixel location using a camera.*
- **VisualPtr GetVisual** (const **math::Vector2i** &_mousePos) const

  *Get a visual at a mouse position.*
- void **HandleKeyPressEvent** (const std::string &_key)

  *Handle a key press.*
- void **HandleKeyReleaseEvent** (const std::string &_key)

  *Handle a key release.*
- void **HandleMouseEvent** (const **common::MouseEvent** &_evt)

  *Handle a mouse event.*
- void **Init** ()

  *Initialize.*
- void **Load** (sdf::ElementPtr _sdf)

  *Load the user camera.*
- void **Load** ()

  *Generic load function.*
- virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)

  *Move the camera to a position (this is an animated motion).*
- void **MoveToVisual** (**VisualPtr** _visual)

  *Move the camera to focus on a visual.*
- void **MoveToVisual** (const std::string &_visualName)

  *Move the camera to focus on a visual.*
- virtual void **PostRender** ()

  *Post render.*
- void **Resize** (unsigned int _w, unsigned int _h)

  *Resize the camera.*
- void **SetFocalPoint** (const **math::Vector3** &_pt)

  *Set the point the camera should orbit around.*
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

  *Set to true to enable rendering.*
- void **SetViewController** (const std::string &_type)

  *Set view controller.*
- void **SetViewController** (const std::string &_type, const **math::Vector3** &_pos)

*Set view controller.*

- void **SetViewportDimensions** (float _x, float _y, float _w, float _h)

  *Set the dimensions of the viewport.*
- virtual void **SetWorldPose** (const **math::Pose** &_pose)

  *Set the pose in the world coordinate frame.*
- virtual void **Update** ()

  *Render the camera.*

### Protected Member Functions

- virtual void **AnimationComplete** ()

  *Internal function used to indicate that an animation has completed.*
- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  *Set the camera to be attached to a visual.*
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

  *Set the camera to track a scene node.*

### Additional Inherited Members

## 10.220.1 Detailed Description

A camera used for user visualization of a scene.

## 10.220.2 Constructor & Destructor Documentation

#### 10.220.2.1 gazebo::rendering::UserCamera::UserCamera ( const std::string & *_name,* ScenePtr *_scene* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the camera. |
| in | *_scene* | **Scene** (p. 814) to put the camera in. |

#### 10.220.2.2 virtual gazebo::rendering::UserCamera::~UserCamera ( ) [virtual]

Destructor.

## 10.220.3 Member Function Documentation

#### 10.220.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete ( ) [protected], [virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 193).

#### 10.220.3.2 virtual bool gazebo::rendering::UserCamera::AttachToVisualImpl ( VisualPtr _visual, bool _inheritOrientation, double _minDist = 0, double _maxDist = 0 ) [protected], [virtual]

Set the camera to be attached to a visual.

This causes the camera to move in relation to the specified visual.

**Parameters**

| | | |
|---|---|---|
| in | *_visual* | The visual to attach to. |
| in | *_inherit-Orientation* | True if the camera should also rotate when the visual rotates. |
| in | *_minDist* | Minimum distance the camera can get to the visual. |
| in | *_maxDist* | Maximum distance the camera can get from the visual. |

**Returns**

True if successfully attach to the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 194).

#### 10.220.3.3 void gazebo::rendering::UserCamera::EnableViewController ( bool _value ) const

Set whether the view controller is enabled.

The view controller is used to handle user camera movements.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | True to enable viewcontroller, False to disable. |

#### 10.220.3.4 void gazebo::rendering::UserCamera::Fini ( ) [virtual]

Finialize.

Reimplemented from **gazebo::rendering::Camera** (p. 195).

#### 10.220.3.5 float gazebo::rendering::UserCamera::GetAvgFPS ( ) const

Get the average frames per second.

**Returns**

The average rendering frames per second

#### 10.220.3.6 GUIOverlay* gazebo::rendering::UserCamera::GetGUIOverlay ( )

Get the GUI overlay.

An overlay allows you to draw 2D elements on the viewport.

**Returns**

Pointer to the **GUIOverlay** (p. 455).

#### 10.220.3.7 virtual unsigned int gazebo::rendering::UserCamera::GetImageHeight ( ) const [virtual]

Get the height of the image.

**Returns**

Image height

Reimplemented from **gazebo::rendering::Camera** (p. 198).

#### 10.220.3.8 virtual unsigned int gazebo::rendering::UserCamera::GetImageWidth ( ) const [virtual]

Get the width of the image.

**Returns**

Image width

Reimplemented from **gazebo::rendering::Camera** (p. 198).

#### 10.220.3.9 float gazebo::rendering::UserCamera::GetTriangleCount ( ) const

Get the triangle count.

**Returns**

The number of triangles currently being rendered.

#### 10.220.3.10 std::string gazebo::rendering::UserCamera::GetViewControllerTypeString ( )

Get current view controller type.

**Returns**

Type of the current view controller: "orbit", "fps"

#### 10.220.3.11 VisualPtr gazebo::rendering::UserCamera::GetVisual ( const math::Vector2i & *_mousePos*, std::string & *_mod* )

Get an entity at a pixel location using a camera.

Used for mouse picking.

**Parameters**

| | | |
|---|---|---|
| in | *_mousePos* | The position of the mouse in screen coordinates |
| out | *_mod* | Used for object manipulation |

**Returns**

The selected entity, or NULL

#### 10.220.3.12 VisualPtr gazebo::rendering::UserCamera::GetVisual ( const math::Vector2i & *_mousePos* ) const

Get a visual at a mouse position.

**Parameters**

| | | |
|---|---|---|
| in | *_mousePos* | 2D position of the mouse in pixels. |

#### 10.220.3.13 void gazebo::rendering::UserCamera::HandleKeyPressEvent ( const std::string & *_key* )

Handle a key press.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key pressed. |

#### 10.220.3.14 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent ( const std::string & *_key* )

Handle a key release.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key released. |

#### 10.220.3.15 void gazebo::rendering::UserCamera::HandleMouseEvent ( const common::MouseEvent & *_evt* )

Handle a mouse event.

**Parameters**

| | | |
|---|---|---|
| in | *_evt* | The mouse event. |

#### 10.220.3.16 void gazebo::rendering::UserCamera::Init ( ) [virtual]

Initialize.

Reimplemented from **gazebo::rendering::Camera** (p. 203).

#### 10.220.3.17 void gazebo::rendering::UserCamera::Load ( sdf::ElementPtr *_sdf* ) [virtual]

Load the user camera.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Parameters for the camera. |

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.220.3.18 void gazebo::rendering::UserCamera::Load ( ) [virtual]

Generic load function.

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.220.3.19 virtual bool gazebo::rendering::UserCamera::MoveToPosition ( const math::Pose & *_pose*, double *_time* ) [virtual]

Move the camera to a position (this is an animated motion).

**See Also**

**Camera::MoveToPositions** (p. 204)

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | End position of the camera |
| in | *_time* | Duration of the camera's movement |

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.220.3.20 void gazebo::rendering::UserCamera::MoveToVisual ( VisualPtr *_visual* )

Move the camera to focus on a visual.

**Parameters**

| | | |
|---|---|---|
| in | *_visual* | **Visual** (p. 1121) to move the camera to. |

#### 10.220.3.21 void gazebo::rendering::UserCamera::MoveToVisual ( const std::string & *_visualName* )

Move the camera to focus on a visual.

**Parameters**

| | | |
|---|---|---|
| in | *_visualName* | Name of the visual to move the camera to. |

#### 10.220.3.22 virtual void gazebo::rendering::UserCamera::PostRender ( ) [virtual]

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 204).

#### 10.220.3.23 void gazebo::rendering::UserCamera::Resize ( unsigned int _w, unsigned int _h )

Resize the camera.

**Parameters**

| | | |
|---|---|---|
| in | *_w* | Width of the camera image. |
| in | *_h* | Height of the camera image. |

#### 10.220.3.24 void gazebo::rendering::UserCamera::SetFocalPoint ( const math::Vector3 & _pt )

Set the point the camera should orbit around.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The focal point |

#### 10.220.3.25 virtual void gazebo::rendering::UserCamera::SetRenderTarget ( Ogre::RenderTarget * _target ) [virtual]

Set to true to enable rendering.

Use this only if you really know what you're doing.

**Parameters**

| | | |
|---|---|---|
| in | *_target* | The new rendering target. |

Reimplemented from **gazebo::rendering::Camera** (p. 208).

#### 10.220.3.26 void gazebo::rendering::UserCamera::SetViewController ( const std::string & _type )

Set view controller.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | The type of view controller: "orbit", "fps" |

#### 10.220.3.27 void gazebo::rendering::UserCamera::SetViewController ( const std::string & _type, const math::Vector3 & _pos )

Set view controller.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | The type of view controller: "orbit", "fps" |
| in | *_pos* | The initial pose of the camera. |

#### 10.220.3.28 void gazebo::rendering::UserCamera::SetViewportDimensions ( float _x, float _y, float _w, float _h )

Set the dimensions of the viewport.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | X position of the viewport. |
| in | *_y* | Y position of the viewport. |
| in | *_w* | Width of the viewport. |
| in | *_h* | Height of the viewport. |

#### 10.220.3.29 virtual void gazebo::rendering::UserCamera::SetWorldPose ( const math::Pose & *_pose* ) [virtual]

Set the pose in the world coordinate frame.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | New pose of the camera. |

Reimplemented from **gazebo::rendering::Camera** (p. 208).

#### 10.220.3.30 virtual bool gazebo::rendering::UserCamera::TrackVisualImpl ( VisualPtr *_visual* ) [protected], [virtual]

Set the camera to track a scene node.

Tracking just causes the camera to rotate to follow the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_visual* | **Visual** (p. 1121) to track. |

**Returns**

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 210).

#### 10.220.3.31 virtual void gazebo::rendering::UserCamera::Update ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 210).

The documentation for this class was generated from the following file:

- **UserCamera.hh**

## 10.221 gazebo::math::Vector2d Class Reference

Generic double x, y vector.

`#include <Vector2d.hh>`

### Public Member Functions

- **Vector2d** ()

*Constructor.*

- **Vector2d** (const double &_x, const double &_y)

  *Constructor.*
- **Vector2d** (const **Vector2d** &_v)

  *Copy constructor.*
- virtual ∼**Vector2d** ()

  *Destructor.*
- **Vector2d Cross** (const **Vector2d** &_v) const

  *Return the cross product of this vector and _v.*
- double **Distance** (const **Vector2d** &_pt) const

  *Calc distance to the given point.*
- bool **IsFinite** () const

  *See if a point is finite (e.g., not nan)*
- void **Normalize** ()

  *Normalize the vector length.*
- bool **operator!=** (const **Vector2d** &_v) const

  *Not equal to operator.*
- const **Vector2d operator**∗ (const **Vector2d** &_v) const

  *Multiplication operators.*
- const **Vector2d operator**∗ (double _v) const

  *Multiplication operators.*
- const **Vector2d** & **operator**∗**=** (const **Vector2d** &_v)

  *Multiplication assignment operator.*
- const **Vector2d** & **operator**∗**=** (double _v)

  *Multiplication assignment operator.*
- **Vector2d operator+** (const **Vector2d** &_v) const

  *Addition operator.*
- const **Vector2d** & **operator+=** (const **Vector2d** &_v)

  *Addition assignment operator.*
- **Vector2d operator-** (const **Vector2d** &_v) const

  *Subtraction operator.*
- const **Vector2d** & **operator-=** (const **Vector2d** &_v)

  *Subtraction assignment operator.*
- const **Vector2d operator**/ (const **Vector2d** &_v) const

  *Division operator.*
- const **Vector2d operator**/ (double _v) const

  *Division operator.*
- const **Vector2d** & **operator**/**=** (const **Vector2d** &_v)

  *Division operator.*
- const **Vector2d** & **operator**/**=** (double _v)

  *Division operator.*
- **Vector2d** & **operator=** (const **Vector2d** &_v)

  *Assignment operator.*
- const **Vector2d** & **operator=** (double _v)

  *Assignment operator.*
- bool **operator==** (const **Vector2d** &_v) const

  *Equal to operator.*

- double **operator[]** (unsigned int _index) const

  *Array subscript operator.*
- void **Set** (double _x, double _y)

  *Set the contents of the vector.*

### Public Attributes

- double **x**

  *x data*
- double **y**

  *y data*

### Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2d** &_pt)

  *Stream extraction operator.*
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2d** &_pt)

  *Stream extraction operator.*

## 10.221.1 Detailed Description

Generic double x, y vector.

## 10.221.2 Constructor & Destructor Documentation

#### 10.221.2.1 gazebo::math::Vector2d::Vector2d ( )

Constructor.

#### 10.221.2.2 gazebo::math::Vector2d::Vector2d ( const double & *_x*, const double & *_y* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | value along x |
| in | *_y* | value along y |

#### 10.221.2.3 gazebo::math::Vector2d::Vector2d ( const Vector2d & *_v* )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the value |

#### 10.221.2.4 virtual gazebo::math::Vector2d::~Vector2d ( ) [virtual]

Destructor.

### 10.221.3 Member Function Documentation

#### 10.221.3.1 Vector2d gazebo::math::Vector2d::Cross ( const Vector2d & _v ) const

Return the cross product of this vector and _v.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the vector |

**Returns**

the cross product

#### 10.221.3.2 double gazebo::math::Vector2d::Distance ( const Vector2d & _pt ) const

Calc distance to the given point.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The point to measure to |

**Returns**

the distance

#### 10.221.3.3 bool gazebo::math::Vector2d::IsFinite ( ) const

See if a point is finite (e.g., not nan)

**Returns**

true if finite, false otherwise

#### 10.221.3.4 void gazebo::math::Vector2d::Normalize ( )

Normalize the vector length.

#### 10.221.3.5 bool gazebo::math::Vector2d::operator!= ( const Vector2d & _v ) const

Not equal to operator.

**Returns**

true if elements are of diffent values (tolerence 1e-6)

#### 10.221.3.6 const Vector2d gazebo::math::Vector2d::operator∗ ( const Vector2d & _v ) const

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector |

**Returns**

the result

#### 10.221.3.7 const Vector2d gazebo::math::Vector2d::operator∗ ( double _v ) const

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | _v | the scaling factor |

**Returns**

a scaled vector

#### 10.221.3.8 const Vector2d& gazebo::math::Vector2d::operator∗= ( const Vector2d & _v )

Multiplication assignment operator.

**Remarks**

this is an element wise multiplication

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector |

**Returns**

this

#### 10.221.3.9 const Vector2d& gazebo::math::Vector2d::operator∗= ( double _v )

Multiplication assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the scaling factor |

**Returns**

a scaled vector

#### 10.221.3.10 Vector2d gazebo::math::Vector2d::operator+ ( const Vector2d & _v ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | vector to add |

**Returns**

sum vector

#### 10.221.3.11 const Vector2d& gazebo::math::Vector2d::operator+= ( const Vector2d & _v )

Addition assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to add |

#### 10.221.3.12 Vector2d gazebo::math::Vector2d::operator- ( const Vector2d & _v ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to substract |

**Returns**

the subtracted vector

#### 10.221.3.13 const Vector2d& gazebo::math::Vector2d::operator-= ( const Vector2d & _v )

Subtraction assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to substract |

**Returns**

this

#### 10.221.3.14 const Vector2d gazebo::math::Vector2d::operator/ ( const Vector2d & _v ) const

Division operator.

**Remarks**

this is an element wise division

**Parameters**

| | | |
|---|---|---|
| in | _v | a vector |

**Returns**

a result

#### 10.221.3.15 const Vector2d gazebo::math::Vector2d::operator/ ( double _v ) const

Division operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the value |

**Returns**

a vector

#### 10.221.3.16 const Vector2d& gazebo::math::Vector2d::operator/= ( const Vector2d & _v )

Division operator.

**Remarks**

this is an element wise division

**Parameters**

| | | |
|---|---|---|
| in | _v | a vector |

**Returns**

this

#### 10.221.3.17 const Vector2d& gazebo::math::Vector2d::operator/= ( double _v )

Division operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the divisor |

**Returns**

a vector

#### 10.221.3.18 Vector2d& gazebo::math::Vector2d::operator= ( const Vector2d & _v )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | a value for x and y element |

**Returns**

this

#### 10.221.3.19 const Vector2d& gazebo::math::Vector2d::operator= ( double _v )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the value for x and y element |

**Returns**

this

#### 10.221.3.20 bool gazebo::math::Vector2d::operator== ( const Vector2d & _v ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the vector to compare to |

**Returns**

true if the elements of the 2 vectors are equal within a tolerence (1e-6)

#### 10.221.3.21 double gazebo::math::Vector2d::operator[] ( unsigned int *_index* ) const

Array subscript operator.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | the index |

**Returns**

the value, or 0 if _index is out of bounds

#### 10.221.3.22 void gazebo::math::Vector2d::Set ( double *_x*, double *_y* )

Set the contents of the vector.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | value along x |
| in | *_y* | value along y |

### 10.221.4 Friends And Related Function Documentation

#### 10.221.4.1 std::ostream& operator<< ( std::ostream & *_out*, const gazebo::math::Vector2d & *_pt* ) [friend]

Stream extraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_pt* | **Vector2d** (p. 1074) to output |

**Returns**

The stream

#### 10.221.4.2 std::istream& operator>> ( std::istream & *_in*, gazebo::math::Vector2d & *_pt* ) [friend]

Stream extraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_in* | input stream |
| in | *_pt* | **Vector3** (p. 1091) to read values into |

**Returns**

The stream

### 10.221.5 Member Data Documentation

#### 10.221.5.1 double gazebo::math::Vector2d::x

x data

#### 10.221.5.2 double gazebo::math::Vector2d::y

y data

The documentation for this class was generated from the following file:

- **Vector2d.hh**

## 10.222 gazebo::math::Vector2i Class Reference

Generic integer x, y vector.

`#include <math/gzmath.hh>`

### Public Member Functions

- **Vector2i** ()
  *Constructor.*
- **Vector2i** (const int &_x, const int &_y)
  *Constructor.*
- **Vector2i** (const **Vector2i** &_pt)
  *Copy onstructor.*
- virtual ∼**Vector2i** ()
  *Destructor.*
- **Vector2i Cross** (const **Vector2i** &_pt) const
  *Return the cross product of this vector and _pt.*
- int **Distance** (const **Vector2i** &_pt) const
  *Calc distance to the given point.*
- bool **IsFinite** () const
  *See if a point is finite (e.g., not nan)*
- void **Normalize** ()
  *Normalize the vector length.*
- bool **operator!=** (const **Vector2i** &_v) const
  *Equality operators.*
- const **Vector2i operator**∗ (const **Vector2i** &_v) const
  *Multiplication operator.*
- const **Vector2i operator**∗ (int _v) const
  *Multiplication operator.*
- const **Vector2i** & **operator**∗**=** (const **Vector2i** &_v)
  *Multiplication operators.*
- const **Vector2i** & **operator**∗**=** (int _v)
  *Multiplication operator.*
- **Vector2i operator+** (const **Vector2i** &_v) const
  *Addition operator.*
- const **Vector2i** & **operator+=** (const **Vector2i** &_v)
  *Addition assignment operator.*
- **Vector2i operator-** (const **Vector2i** &_v) const
  *Subtraction operator.*

- const **Vector2i** & **operator-=** (const **Vector2i** &_v)

  *Subtraction operators.*
- const **Vector2i operator**/ (const **Vector2i** &_v) const

  *Division operator.*
- const **Vector2i operator**/ (int _v) const

  *Division operator.*
- const **Vector2i** & **operator**/= (const **Vector2i** &_v)

  *Division operator.*
- const **Vector2i** & **operator**/= (int _v)

  *Division operator.*
- **Vector2i** & **operator=** (const **Vector2i** &_v)

  *Assignment operator.*
- const **Vector2i** & **operator=** (int _value)

  *Assignment operator.*
- bool **operator==** (const **Vector2i** &_v) const

  *Equality operator.*
- int **operator[]** (unsigned int _index) const

  *Array subscript operator.*
- void **Set** (int _x, int _y)

  *Set the contents of the vector.*

## Public Attributes

- int **x**

  *x data*
- int **y**

  *y data*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2i** &_pt)

  *Stream insertion operator.*
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2i** &_pt)

  *Stream extraction operator.*

### 10.222.1 Detailed Description

Generic integer x, y vector.

### 10.222.2 Constructor & Destructor Documentation

#### 10.222.2.1 gazebo::math::Vector2i::Vector2i ( )

Constructor.

#### 10.222.2.2 gazebo::math::Vector2i::Vector2i ( const int & _x, const int & _y )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | value along x |
| in | *_y* | value along y |

#### 10.222.2.3 gazebo::math::Vector2i::Vector2i ( const Vector2i & _pt )

Copy onstructor.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | a point |

#### 10.222.2.4 virtual gazebo::math::Vector2i::~Vector2i ( ) [virtual]

Destructor.

### 10.222.3 Member Function Documentation

#### 10.222.3.1 Vector2i gazebo::math::Vector2i::Cross ( const Vector2i & _pt ) const

Return the cross product of this vector and _pt.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | the other vector |

**Returns**

the product

#### 10.222.3.2 int gazebo::math::Vector2i::Distance ( const Vector2i & _pt ) const

Calc distance to the given point.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | a point |

**Returns**

the distance

#### 10.222.3.3 bool gazebo::math::Vector2i::IsFinite ( ) const

See if a point is finite (e.g., not nan)

**Returns**

the result

#### 10.222.3.4 void gazebo::math::Vector2i::Normalize ( )

Normalize the vector length.

#### 10.222.3.5 bool gazebo::math::Vector2i::operator!= ( const Vector2i & _v ) const

Equality operators.

**Parameters**

| | |
|---|---|
| _v | the vector to compare with |

**Returns**

true if component have different values, false otherwise

#### 10.222.3.6 const Vector2i gazebo::math::Vector2i::operator* ( const Vector2i & _v ) const

Multiplication operator.

**Remarks**

this is an element wise multiplication

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector |

**Returns**

the result

#### 10.222.3.7 const Vector2i gazebo::math::Vector2i::operator* ( int _v ) const

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the scaling factor |

**Returns**

the result

#### 10.222.3.8 const Vector2i& gazebo::math::Vector2i::operator*= ( const Vector2i & _v )

Multiplication operators.

**Remarks**

this is an element wise multiplication

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector |

**Returns**

this

#### 10.222.3.9 const Vector2i& gazebo::math::Vector2i::operator*= ( int _v )

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | scaling factor |

**Returns**

this

#### 10.222.3.10 Vector2i gazebo::math::Vector2i::operator+ ( const Vector2i & _v ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to add |

**Returns**

the sum vector

#### 10.222.3.11 const Vector2i& gazebo::math::Vector2i::operator+= ( const Vector2i & _v )

Addition assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to add |

**Returns**

this

#### 10.222.3.12 Vector2i gazebo::math::Vector2i::operator- ( const Vector2i & _v ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to substract |

**Returns**

the result vector

#### 10.222.3.13 const Vector2i& gazebo::math::Vector2i::operator-= ( const Vector2i & _v )

Subtraction operators.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to substract |

**Returns**

this

#### 10.222.3.14 const Vector2i gazebo::math::Vector2i::operator/ ( const Vector2i & _v ) const

Division operator.

**Remarks**

this is an element wise division.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to divide |

**Returns**

the result

#### 10.222.3.15 const Vector2i gazebo::math::Vector2i::operator/ ( int _v ) const

Division operator.

**Remarks**

this is an element wise division.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to divide |

**Returns**

the result

#### 10.222.3.16 const Vector2i& gazebo::math::Vector2i::operator/= ( const Vector2i & _v )

Division operator.

**Remarks**

this is an element wise division.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to divide |

**Returns**

this

#### 10.222.3.17 const Vector2i& gazebo::math::Vector2i::operator/= ( int _v )

Division operator.

**Remarks**

this is an element wise division.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to divide |

**Returns**

this

#### 10.222.3.18 Vector2i& gazebo::math::Vector2i::operator= ( const Vector2i & _v )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the value |

**Returns**

this

#### 10.222.3.19 const Vector2i& gazebo::math::Vector2i::operator= ( int _value )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | _value | the value for x and y |

**Returns**

this

#### 10.222.3.20 bool gazebo::math::Vector2i::operator== ( const Vector2i & _v ) const

Equality operator.

**Parameters**

| | |
|---|---|
| _v | the vector to compare with |

**Returns**

true if component have the same values, false otherwise

#### 10.222.3.21 int gazebo::math::Vector2i::operator[] ( unsigned int _index ) const

Array subscript operator.

**Parameters**

| | | |
|---|---|---|
| in | _index | the array index |

#### 10.222.3.22 void gazebo::math::Vector2i::Set ( int _x, int _y )

Set the contents of the vector.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | value along x |
| in | *_y* | value along y |

### 10.222.4 Friends And Related Function Documentation

#### 10.222.4.1 std::ostream& operator<< ( std::ostream & *_out,* const gazebo::math::Vector2i & *_pt* ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *pt* | **Vector2i** (p. 1083) to output |

**Returns**

the stream

#### 10.222.4.2 std::istream& operator>> ( std::istream & *_in,* gazebo::math::Vector2i & *_pt* ) [friend]

Stream extraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_in* | input stream |
| in | *_pt* | **Vector3** (p. 1091) to read values into |

**Returns**

The stream

### 10.222.5 Member Data Documentation

#### 10.222.5.1 int gazebo::math::Vector2i::x

x data

#### 10.222.5.2 int gazebo::math::Vector2i::y

y data

The documentation for this class was generated from the following file:

- **Vector2i.hh**

## 10.223 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 1091) class represents the generic vector containing 3 elements.

`#include <math/gzmath.hh>`

## Public Member Functions

- **Vector3** ()

  *Constructor.*
- **Vector3** (const double &_x, const double &_y, const double &_z)

  *Constructor.*
- **Vector3** (const **Vector3** &_v)

  *Copy constructor.*
- virtual ~**Vector3** ()

  *Destructor.*
- void **Correct** ()

  *Corrects any nan values.*
- **Vector3 Cross** (const **Vector3** &_pt) const

  *Return the cross product of this vector and pt.*
- double **Distance** (const **Vector3** &_pt) const

  *Calc distance to the given point.*
- double **Distance** (double _x, double _y, double _z) const

  *Calc distance to the given point.*
- double **Dot** (const **Vector3** &_pt) const

  *Return the dot product of this vector and pt.*
- bool **Equal** (const **Vector3** &_v) const

  *Equality test.*
- **Vector3 GetAbs** () const

  *Get the absolute value of the vector.*
- double **GetDistToLine** (const **Vector3** &_pt1, const **Vector3** &_pt2)

  *Get distance to a line.*
- double **GetLength** () const

  *Returns the length (magnitude) of the vector \ return the length.*
- double **GetMax** () const

  *Get the maximum value in the vector.*
- double **GetMin** () const

  *Get the minimum value in the vector.*
- **Vector3 GetPerpendicular** () const

  *Return a vector that is perpendicular to this one.*
- **Vector3 GetRounded** () const

  *Get a rounded version of this vector.*
- double **GetSquaredLength** () const

  *Return the square of the length (magnitude) of the vector.*
- double **GetSum** () const

  *Return the sum of the values.*
- bool **IsFinite** () const

  *See if a point is finite (e.g., not nan)*
- **Vector3 Normalize** ()

  *Normalize the vector length.*
- bool **operator!=** (const **Vector3** &_v) const

*Not equal to operator.*

- **Vector3 operator**∗ (const **Vector3** &_p) const

  *Multiplication operator.*

- **Vector3 operator**∗ (double _v) const

  *Multiplication operators.*

- const **Vector3** & **operator∗=** (const **Vector3** &_v)

  *Multiplication operators.*

- const **Vector3** & **operator∗=** (double _v)

  *Multiplication operator.*

- **Vector3 operator+** (const **Vector3** &_v) const

  *Addition operator.*

- const **Vector3** & **operator+=** (const **Vector3** &_v)

  *Addition assignment operator.*

- **Vector3 operator-** () const

  *Negation operator.*

- **Vector3 operator-** (const **Vector3** &_pt) const

  *Subtraction operators.*

- const **Vector3** & **operator-=** (const **Vector3** &_pt)

  *Subtraction operators.*

- const **Vector3 operator**/ (const **Vector3** &_pt) const

  *Division operator.*

- const **Vector3 operator**/ (double _v) const

  *Division operator.*

- const **Vector3** & **operator**/**=** (const **Vector3** &_pt)

  *Division assignment operator.*

- const **Vector3** & **operator**/**=** (double _v)

  *Division operator.*

- **Vector3** & **operator=** (const **Vector3** &_v)

  *Assignment operator.*

- **Vector3** & **operator=** (double _value)

  *Assignment operator.*

- bool **operator==** (const **Vector3** &_pt) const

  *Equal to operator.*

- double **operator[ ]** (unsigned int index) const

  *[] operator*

- **Vector3 Round** ()

  *Round to near whole number, return the result.*

- void **Round** (int _precision)

  *Round all values to _precision decimal places.*

- void **Set** (double _x=0, double _y=0, double _z=0)

  *Set the contents of the vector.*

- void **SetToMax** (const **Vector3** &_v)

  *Set this vector's components to the maximum of itself and the passed in vector.*

- void **SetToMin** (const **Vector3** &_v)

  *Set this vector's components to the minimum of itself and the passed in vector.*

### Static Public Member Functions

- static **Vector3 GetNormal** (const **Vector3** &_v1, const **Vector3** &_v2, const **Vector3** &_v3)
  *Get a normal vector to a triangle.*

### Public Attributes

- double **x**
  *X location.*
- double **y**
  *Y location.*
- double **z**
  *Z location.*

### Static Public Attributes

- static const **Vector3 One**
  *math::Vector3(1, 1, 1)*
- static const **Vector3 UnitX**
  *math::Vector3(1, 0, 0)*
- static const **Vector3 UnitY**
  *math::Vector3(0, 1, 0)*
- static const **Vector3 UnitZ**
  *math::Vector3(0, 0, 1)*
- static const **Vector3 Zero**
  *math::Vector3(0, 0, 0)*

### Friends

- **Vector3 operator**∗ (double _s, const **Vector3** &_v)
  *Multiplication operators.*
- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector3** &_pt)
  *Stream insertion operator.*
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector3** &_pt)
  *Stream extraction operator.*

## 10.223.1 Detailed Description

The **Vector3** (p. 1091) class represents the generic vector containing 3 elements.

Since it's commonly used to keep coordinate system related information, its elements are labeled by x, y, z.

## 10.223.2 Constructor & Destructor Documentation

#### 10.223.2.1 gazebo::math::Vector3::Vector3 ( )

Constructor.

Referenced by operator-().

#### 10.223.2.2 gazebo::math::Vector3::Vector3 ( const double & _x, const double & _y, const double & _z )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | _x | value along x |
| in | _y | value along y |
| in | _z | value along z |

#### 10.223.2.3 gazebo::math::Vector3::Vector3 ( const Vector3 & _v )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | _v | a vector |

#### 10.223.2.4 virtual gazebo::math::Vector3::~Vector3 ( ) [virtual]

Destructor.

### 10.223.3 Member Function Documentation

#### 10.223.3.1 void gazebo::math::Vector3::Correct ( ) [inline]

Corrects any nan values.

References x, y, and z.

Referenced by gazebo::math::Pose::Correct().

#### 10.223.3.2 Vector3 gazebo::math::Vector3::Cross ( const Vector3 & _pt ) const

Return the cross product of this vector and pt.

**Returns**

the product

#### 10.223.3.3 double gazebo::math::Vector3::Distance ( const Vector3 & _pt ) const

Calc distance to the given point.

**Parameters**

| | | |
|---|---|---|
| in | _pt | the point |

**Returns**

the distance

#### 10.223.3.4 double gazebo::math::Vector3::Distance ( double _x, double _y, double _z ) const

Calc distance to the given point.

**Parameters**

| | | |
|---|---|---|
| in | _x | value along x |
| in | _y | value along y |
| in | _z | value along z |

**Returns**

the distance

#### 10.223.3.5 double gazebo::math::Vector3::Dot ( const Vector3 & _pt ) const

Return the dot product of this vector and pt.

**Returns**

the product

#### 10.223.3.6 bool gazebo::math::Vector3::Equal ( const Vector3 & _v ) const

Equality test.

**Remarks**

This is equivalent to the == operator

**Parameters**

| | | |
|---|---|---|
| in | _v | the other vector |

**Returns**

true if the 2 vectors have the same values, false otherwise

#### 10.223.3.7 Vector3 gazebo::math::Vector3::GetAbs ( ) const

Get the absolute value of the vector.

**Returns**

a vector with positive elements

#### 10.223.3.8 double gazebo::math::Vector3::GetDistToLine ( const Vector3 & _pt1, const Vector3 & _pt2 )

Get distance to a line.

**Parameters**

| | | |
|---|---|---|
| in | *_pt1* | first point on the line |
| in | *_pt2* | second point on the line |

**Returns**

the minimum distance from this point to the line

#### 10.223.3.9 double gazebo::math::Vector3::GetLength ( ) const

Returns the length (magnitude) of the vector \ return the length.

#### 10.223.3.10 double gazebo::math::Vector3::GetMax ( ) const

Get the maximum value in the vector.

**Returns**

the maximum element

#### 10.223.3.11 double gazebo::math::Vector3::GetMin ( ) const

Get the minimum value in the vector.

**Returns**

the minimum element

#### 10.223.3.12 static Vector3 gazebo::math::Vector3::GetNormal ( const Vector3 & _v1, const Vector3 & _v2, const Vector3 & _v3 ) [static]

Get a normal vector to a triangle.

**Parameters**

| | | |
|---|---|---|
| in | *_v1* | first vertex of the triangle |
| in | *_v2* | second vertex |
| in | *_v3* | third vertex |

**Returns**

the normal

#### 10.223.3.13 Vector3 gazebo::math::Vector3::GetPerpendicular ( ) const

Return a vector that is perpendicular to this one.

**Returns**

an orthogonal vector

#### 10.223.3.14 Vector3 gazebo::math::Vector3::GetRounded ( ) const

Get a rounded version of this vector.

**Returns**

a rounded vector

#### 10.223.3.15 double gazebo::math::Vector3::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

**Returns**

the squared length

#### 10.223.3.16 double gazebo::math::Vector3::GetSum ( ) const

Return the sum of the values.

**Returns**

the sum

#### 10.223.3.17 bool gazebo::math::Vector3::IsFinite ( ) const

See if a point is finite (e.g., not nan)

#### 10.223.3.18 Vector3 gazebo::math::Vector3::Normalize ( )

Normalize the vector length.

**Returns**

unit length vector

#### 10.223.3.19 bool gazebo::math::Vector3::operator!= ( const Vector3 & _v ) const

Not equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | The vector to compare against |

**Returns**

true if each component is equal withing a default tolerence (1e-6), false otherwise

#### 10.223.3.20 Vector3 gazebo::math::Vector3::operator∗ ( const Vector3 & _p ) const

Multiplication operator.

**Remarks**

this is an element wise multiplication, not a cross product

**Parameters**

| | | |
|---|---|---|
| in | _v | |

#### 10.223.3.21 Vector3 gazebo::math::Vector3::operator∗ ( double _v ) const

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | _v | the scaling factor |

**Returns**

a scaled vector

#### 10.223.3.22 const Vector3& gazebo::math::Vector3::operator∗= ( const Vector3 & _v )

Multiplication operators.

**Remarks**

this is an element wise multiplication, not a cross product

**Parameters**

| | | |
|---|---|---|
| in | _v | a vector |

**Returns**

this

#### 10.223.3.23 const Vector3& gazebo::math::Vector3::operator*= ( double _v )

Multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | scaling factor |

**Returns**

this

#### 10.223.3.24 Vector3 gazebo::math::Vector3::operator+ ( const Vector3 & _v ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | vector to add |

**Returns**

the sum vector

#### 10.223.3.25 const Vector3& gazebo::math::Vector3::operator+= ( const Vector3 & _v )

Addition assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | vector to add |

#### 10.223.3.26 Vector3 gazebo::math::Vector3::operator- ( ) const [inline]

Negation operator.

**Returns**

negative of this vector

References Vector3(), x, y, and z.

#### 10.223.3.27 Vector3 gazebo::math::Vector3::operator- ( const Vector3 & _pt ) const [inline]

Subtraction operators.

**Parameters**

| | | |
|---|---|---|
| in | _pt | a vector to substract |

**Returns**

a vector

References Vector3(), x, y, and z.

#### 10.223.3.28 const Vector3& gazebo::math::Vector3::operator-= ( const Vector3 & _pt )

Subtraction operators.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | subtrahend |

#### 10.223.3.29 const Vector3 gazebo::math::Vector3::operator/ ( const Vector3 & _pt ) const

Division operator.

[in] _pt the vector divisor

**Remarks**

this is an element wise division

**Returns**

a vector

#### 10.223.3.30 const Vector3 gazebo::math::Vector3::operator/ ( double _v ) const

Division operator.

**Remarks**

this is an element wise division

**Returns**

a vector

#### 10.223.3.31 const Vector3& gazebo::math::Vector3::operator/= ( const Vector3 & _pt )

Division assignment operator.

[in] _pt the vector divisor

**Remarks**

this is an element wise division

**Returns**

a vector

#### 10.223.3.32 const Vector3& gazebo::math::Vector3::operator/= ( double _v )

Division operator.

**Remarks**

this is an element wise division

**Returns**

this

#### 10.223.3.33 Vector3& gazebo::math::Vector3::operator= ( const Vector3 & _v )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | a new value |

**Returns**

this

#### 10.223.3.34 Vector3& gazebo::math::Vector3::operator= ( double _value )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | assigned to all elements |

**Returns**

this

#### 10.223.3.35 bool gazebo::math::Vector3::operator== ( const Vector3 & _pt ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The vector to compare against |

**Returns**

true if each component is equal withing a default tolerence (1e-6), false otherwise

#### 10.223.3.36 double gazebo::math::Vector3::operator[] ( unsigned int *index* ) const

[] operator

#### 10.223.3.37 Vector3 gazebo::math::Vector3::Round ( )

Round to near whole number, return the result.

**Returns**

the result

#### 10.223.3.38 void gazebo::math::Vector3::Round ( int *_precision* )

Round all values to _precision decimal places.

**Parameters**

| | | |
|---|---|---|
| in | *_precision* | the decimal places |

#### 10.223.3.39 void gazebo::math::Vector3::Set ( double *_x* = 0, double *_y* = 0, double *_z* = 0 ) [inline]

Set the contents of the vector.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | value along x |
| in | *_y* | value along y |
| in | *_z* | value aling z |

References x, y, and z.

#### 10.223.3.40 void gazebo::math::Vector3::SetToMax ( const Vector3 & *_v* )

Set this vector's components to the maximum of itself and the passed in vector.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the maximum clamping vector |

#### 10.223.3.41 void gazebo::math::Vector3::SetToMin ( const Vector3 & *_v* )

Set this vector's components to the minimum of itself and the passed in vector.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the minimum clamping vector |

### 10.223.4 Friends And Related Function Documentation

#### 10.223.4.1 Vector3 operator∗ ( double _s, const Vector3 & _v ) [friend]

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | _s | the scaling factor |
| in | _v | input vector |

**Returns**

a scaled vector

#### 10.223.4.2 std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Vector3 & _pt ) [friend]

Stream insertion operator.

**Parameters**

| | |
|---|---|
| _out | output stream |
| _pt | **Vector3** (p. 1091) to output |

**Returns**

the stream

#### 10.223.4.3 std::istream& operator>> ( std::istream & _in, gazebo::math::Vector3 & _pt ) [friend]

Stream extraction operator.

**Parameters**

| | |
|---|---|
| _in | input stream |
| _pt | vector3 to read values into |

**Returns**

the stream

### 10.223.5 Member Data Documentation

#### 10.223.5.1 const Vector3 gazebo::math::Vector3::One [static]

math::Vector3(1, 1, 1)

#### 10.223.5.2 const Vector3 gazebo::math::Vector3::UnitX [static]

math::Vector3(1, 0, 0)

#### 10.223.5.3 const Vector3 gazebo::math::Vector3::UnitY [static]

math::Vector3(0, 1, 0)

#### 10.223.5.4 const Vector3 gazebo::math::Vector3::UnitZ [static]

math::Vector3(0, 0, 1)

#### 10.223.5.5 double gazebo::math::Vector3::x

X location.

Referenced by gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

#### 10.223.5.6 double gazebo::math::Vector3::y

Y location.

Referenced by gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

#### 10.223.5.7 double gazebo::math::Vector3::z

Z location.

Referenced by gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

#### 10.223.5.8 const Vector3 gazebo::math::Vector3::Zero [static]

math::Vector3(0, 0, 0)

The documentation for this class was generated from the following file:

- **Vector3.hh**

## 10.224 gazebo::math::Vector4 Class Reference

double Generic x, y, z, w vector

`#include <math/gzmath.hh>`

### Public Member Functions

- **Vector4** ()

  *Constructor.*

- **Vector4** (const double &_x, const double &_y, const double &_z, const double &_w)

  *Constructor with component values.*
- **Vector4** (const **Vector4** &_v)

  *Copy constructor.*
- virtual ~**Vector4** ()

  *Destructor.*
- double **Distance** (const **Vector4** &_pt) const

  *Calc distance to the given point.*
- double **GetLength** () const

  *Returns the length (magnitude) of the vector.*
- double **GetSquaredLength** () const

  *Return the square of the length (magnitude) of the vector.*
- bool **IsFinite** () const

  *See if a point is finite (e.g., not nan)*
- void **Normalize** ()

  *Normalize the vector length.*
- bool **operator!=** (const **Vector4** &_pt) const

  *Not equal to operator.*
- const **Vector4** **operator*** (const **Vector4** &_pt) const

  *Multiplication operator.*
- const **Vector4** **operator*** (const **Matrix4** &_m) const

  *Matrix multiplication operator.*
- const **Vector4** **operator*** (double _v) const

  *Multiplication operators.*
- const **Vector4** & **operator*=** (const **Vector4** &_pt)

  *Multiplication assignment operator.*
- const **Vector4** & **operator*=** (double _v)

  *Multiplication assignment operator.*
- **Vector4** **operator+** (const **Vector4** &_v) const

  *Addition operator.*
- const **Vector4** & **operator+=** (const **Vector4** &_v)

  *Addition operator.*
- **Vector4** **operator-** (const **Vector4** &_v) const

  *Subtraction operator.*
- const **Vector4** & **operator-=** (const **Vector4** &_v)

  *Subtraction assigment operators.*
- const **Vector4** **operator**/ (const **Vector4** &_v) const

  *Division assignment operator.*
- const **Vector4** **operator**/ (double _v) const

  *Division assignment operator.*
- const **Vector4** & **operator**/= (const **Vector4** &_v)

  *Division assignment operator.*
- const **Vector4** & **operator**/= (double _v)

  *Division operator.*
- **Vector4** & **operator=** (const **Vector4** &_v)

  *Assignment operator.*
- **Vector4** & **operator=** (double _value)

*Assignment operator.*

- bool **operator==** (const **Vector4** &_pt) const

  *Equal to operator.*

- double **operator[]** (unsigned int _index) const

  *Array subscript operator.*

- void **Set** (double _x=0, double _y=0, double _z=0, double _w=0)

  *Set the contents of the vector.*

## Public Attributes

- double **w**

  *W value.*

- double **x**

  *X value.*

- double **y**

  *Y value.*

- double **z**

  *Z value.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector4** &_pt)

  *Stream insertion operator.*

- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector4** &_pt)

  *Stream extraction operator.*

### 10.224.1 Detailed Description

double Generic x, y, z, w vector

### 10.224.2 Constructor & Destructor Documentation

#### 10.224.2.1 gazebo::math::Vector4::Vector4 ( )

Constructor.

#### 10.224.2.2 gazebo::math::Vector4::Vector4 ( const double & _x, const double & _y, const double & _z, const double & _w )

Constructor with component values.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | value along x axis |
| in | *_y* | value along y axis |
| in | *_z* | value along z axis |
| in | *_w* | value along w axis |

#### 10.224.2.3 gazebo::math::Vector4::Vector4 ( const Vector4 & _v )

Copy constructor.

**Parameters**

| | | |
|---|---|---|
| in | _v | vector |

#### 10.224.2.4 virtual gazebo::math::Vector4::~Vector4 ( ) [virtual]

Destructor.

### 10.224.3 Member Function Documentation

#### 10.224.3.1 double gazebo::math::Vector4::Distance ( const Vector4 & _pt ) const

Calc distance to the given point.

**Parameters**

| | | |
|---|---|---|
| in | _pt | the point |

**Returns**

the distance

#### 10.224.3.2 double gazebo::math::Vector4::GetLength ( ) const

Returns the length (magnitude) of the vector.

#### 10.224.3.3 double gazebo::math::Vector4::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

**Returns**

the length

#### 10.224.3.4 bool gazebo::math::Vector4::IsFinite ( ) const

See if a point is finite (e.g., not nan)

**Returns**

true if finite, false otherwise

#### 10.224.3.5 void gazebo::math::Vector4::Normalize ( )

Normalize the vector length.

#### 10.224.3.6 bool gazebo::math::Vector4::operator!= ( const Vector4 & _pt ) const

Not equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | the other vector |

**Returns**

true if each component is equal withing a default tolerence (1e-6), false otherwise

#### 10.224.3.7 const Vector4 gazebo::math::Vector4::operator* ( const Vector4 & _pt ) const

Multiplication operator.

**Remarks**

Performs element wise multiplication, which has limited use.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | another vector |

**Returns**

result vector

#### 10.224.3.8 const Vector4 gazebo::math::Vector4::operator* ( const Matrix4 & _m ) const

Matrix multiplication operator.

**Parameters**

| | | |
|---|---|---|
| in | *_m* | matrix |

**Returns**

the vector multiplied by _m

#### 10.224.3.9 const Vector4 gazebo::math::Vector4::operator* ( double _v ) const

Multiplication operators.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | scaling factor |

**Returns**

a scaled vector

#### 10.224.3.10 const Vector4& gazebo::math::Vector4::operator*= ( const Vector4 & _pt )

Multiplication assignment operator.

**Remarks**

Performs element wise multiplication, which has limited use.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | a vector |

**Returns**

this

#### 10.224.3.11 const Vector4& gazebo::math::Vector4::operator*= ( double _v )

Multiplication assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | scaling factor |

**Returns**

this

#### 10.224.3.12 Vector4 gazebo::math::Vector4::operator+ ( const Vector4 & _v ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the vector to add |

**Returns**

a sum vector

#### 10.224.3.13 const Vector4& gazebo::math::Vector4::operator+= ( const Vector4 & _v )

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to add |

**Returns**

this vector

#### 10.224.3.14 Vector4 gazebo::math::Vector4::operator- ( const Vector4 & _v ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to substract |

**Returns**

a vector

#### 10.224.3.15 const Vector4& gazebo::math::Vector4::operator-= ( const Vector4 & _v )

Subtraction assigment operators.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to substract |

**Returns**

this vector

#### 10.224.3.16 const Vector4 gazebo::math::Vector4::operator/ ( const Vector4 & _v ) const

Division assignment operator.

**Remarks**

Performs element wise division, which has limited use.

**Parameters**

| | | |
|---|---|---|
| in | _v | the vector to perform element wise division with |

**Returns**

a result vector

#### 10.224.3.17 const Vector4 gazebo::math::Vector4::operator/ ( double _v ) const

Division assignment operator.

**Remarks**

Performs element wise division, which has limited use.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | another vector |

**Returns**

a result vector

#### 10.224.3.18 const Vector4& gazebo::math::Vector4::operator/= ( const Vector4 & _v )

Division assignment operator.

**Remarks**

Performs element wise division, which has limited use.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the vector to perform element wise division with |

**Returns**

this

#### 10.224.3.19 const Vector4& gazebo::math::Vector4::operator/= ( double _v )

Division operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | scaling factor |

**Returns**

a vector

#### 10.224.3.20 Vector4& gazebo::math::Vector4::operator= ( const Vector4 & _v )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_v* | the vector |

**Returns**

a reference to this vector

#### 10.224.3.21 Vector4& gazebo::math::Vector4::operator= ( double *_value* )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | |

#### 10.224.3.22 bool gazebo::math::Vector4::operator== ( const Vector4 & *_pt* ) const

Equal to operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | the other vector |

**Returns**

true if each component is equal withing a default tolerence (1e-6), false otherwise

#### 10.224.3.23 double gazebo::math::Vector4::operator[] ( unsigned int *_index* ) const

Array subscript operator.

**Parameters**

| | | |
|---|---|---|
| in | *_index* | |

#### 10.224.3.24 void gazebo::math::Vector4::Set ( double *_x* = 0, double *_y* = 0, double *_z* = 0, double *_w* = 0 )

Set the contents of the vector.

**Parameters**

| | | |
|---|---|---|
| in | *_x* | value along x axis |
| in | *_y* | value along y axis |
| in | *_z* | value along z axis |
| in | *_w* | value along w axis |

### 10.224.4 Friends And Related Function Documentation

#### 10.224.4.1 std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Vector4 & _pt ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_pt* | **Vector4** (p. 1105) to output |

**Returns**

The stream

#### 10.224.4.2 std::istream& operator>> ( std::istream & _in, gazebo::math::Vector4 & _pt ) [friend]

Stream extraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_in* | input stream |
| in | *_pt* | **Vector4** (p. 1105) to read values into |

**Returns**

the stream

### 10.224.5 Member Data Documentation

#### 10.224.5.1 double gazebo::math::Vector4::w

W value.

#### 10.224.5.2 double gazebo::math::Vector4::x

X value.

#### 10.224.5.3 double gazebo::math::Vector4::y

Y value.

#### 10.224.5.4 double gazebo::math::Vector4::z

Z value.

The documentation for this class was generated from the following file:

- **Vector4.hh**

## 10.225 gazebo::common::Video Class Reference

Handle video encoding and decoding using libavcodec.

`#include <common/common.hh>`

### Public Member Functions

- **Video** ()

  *Constructor.*
- virtual ∼**Video** ()

  *Destructor.*
- int **GetHeight** () const

  *Get the height of the video in pixels.*
- bool **GetNextFrame** (unsigned char ∗∗_buffer)

  *Get the next frame of the video.*
- int **GetWidth** () const

  *Get the width of the video in pixels.*
- bool **Load** (const std::string &_filename)

  *Load a video file.*

### 10.225.1 Detailed Description

Handle video encoding and decoding using libavcodec.

### 10.225.2 Constructor & Destructor Documentation

#### 10.225.2.1 gazebo::common::Video::Video ( )

Constructor.

#### 10.225.2.2 virtual gazebo::common::Video::∼Video ( ) [virtual]

Destructor.

### 10.225.3 Member Function Documentation

#### 10.225.3.1 int gazebo::common::Video::GetHeight ( ) const

Get the height of the video in pixels.

**Returns**

the height

#### 10.225.3.2 bool gazebo::common::Video::GetNextFrame ( unsigned char ** *_buffer* )

Get the next frame of the video.

**Parameters**

| | | |
|---|---|---|
| out | *_img* | **Image** (p. 474) in which the frame is stored |

**Returns**

false if HAVE_FFMPEG is not defined, true otherwise

#### 10.225.3.3 int gazebo::common::Video::GetWidth ( ) const

Get the width of the video in pixels.

**Returns**

the width

#### 10.225.3.4 bool gazebo::common::Video::Load ( const std::string & *_filename* )

Load a video file.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | Full path of the video file |

**Returns**

false if HAVE_FFMPEG is not defined or if a video stream can't be found

The documentation for this class was generated from the following file:

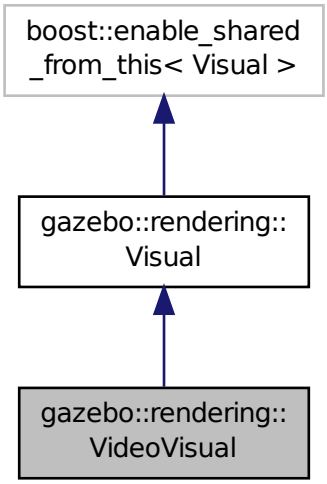
- **Video.hh**

## 10.226 gazebo::rendering::VideoVisual Class Reference

A visual element that displays a video as a texture.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::VideoVisual:




### Public Member Functions

- **VideoVisual** (const std::string &_name, **VisualPtr** _parent)

  *Constructor.*
- virtual ~**VideoVisual** ()

  *Destructor.*

### Additional Inherited Members

## 10.226.1 Detailed Description

A visual element that displays a video as a texture.

## 10.226.2 Constructor & Destructor Documentation

#### 10.226.2.1 gazebo::rendering::VideoVisual::VideoVisual ( const std::string & *_name*, VisualPtr *_parent* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the video visual. |
| in | *_parent* | Parent of the video visual. |

#### 10.226.2.2 virtual gazebo::rendering::VideoVisual::~VideoVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

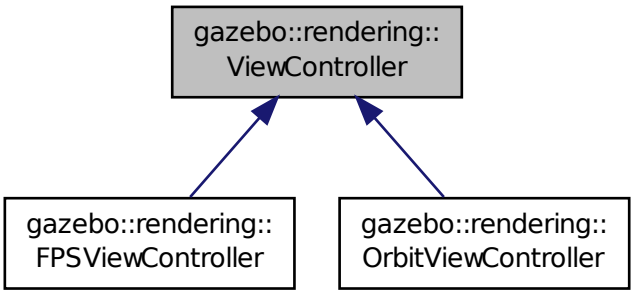
- **VideoVisual.hh**

## 10.227 gazebo::rendering::ViewController Class Reference

Base class for view controllers.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ViewController:




### Public Member Functions

- **ViewController** (**UserCameraPtr** _camera)

  *Constructor.*
- virtual ~**ViewController** ()

  *Destructor.*
- std::string **GetTypeString** () const

  *Get the type of view controller.*
- virtual void **HandleKeyPressEvent** (const std::string &_key)=0

  *Handle a key press event.*
- virtual void **HandleKeyReleaseEvent** (const std::string &_key)=0

  *Handle a key release event.*
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)=0

  *Handle a mouse event.*
- virtual void **Init** ()=0

  *Initialize the view controller.*
- virtual void **Init** (const **math::Vector3** &_focalPoint)

*Initialize with a focus point.*

- void **SetEnabled** (bool _value)

  *Set whether the controller is enabled.*

- virtual void **Update** ()=0

  *Update the controller, which should update the position of the **Camera** (p. 186).*

### Protected Attributes

- **UserCameraPtr camera**

  *Pointer to the camera to control.*

- bool **enabled**

  *True if enabled.*

- std::string **typeString**

  *Type of view controller.*

## 10.227.1 Detailed Description

Base class for view controllers.

## 10.227.2 Constructor & Destructor Documentation

### 10.227.2.1 gazebo::rendering::ViewController::ViewController ( UserCameraPtr *_camera* )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_camera* | The user camera to controll. |

### 10.227.2.2 virtual gazebo::rendering::ViewController::~ViewController ( ) [virtual]

Destructor.

## 10.227.3 Member Function Documentation

### 10.227.3.1 std::string gazebo::rendering::ViewController::GetTypeString ( ) const

Get the type of view controller.

**Returns**

The view controller type string.

### 10.227.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent ( const std::string & *_key* ) [pure virtual]

Handle a key press event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key that was pressed. |

Implemented in **gazebo::rendering::OrbitViewController** (p. 705), and **gazebo::rendering::FPSViewController** (p. 424).

#### 10.227.3.3 virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent ( const std::string & *_key* ) [pure virtual]

Handle a key release event.

**Parameters**

| | | |
|---|---|---|
| in | *_key* | The key that was released. |

Implemented in **gazebo::rendering::OrbitViewController** (p. 705), and **gazebo::rendering::FPSViewController** (p. 424).

#### 10.227.3.4 virtual void gazebo::rendering::ViewController::HandleMouseEvent ( const **common::MouseEvent** & *_event* ) [pure virtual]

Handle a mouse event.

**Parameters**

| | | |
|---|---|---|
| in | *_event* | The mouse position. |

Implemented in **gazebo::rendering::OrbitViewController** (p. 705), and **gazebo::rendering::FPSViewController** (p. 424).

#### 10.227.3.5 virtual void gazebo::rendering::ViewController::Init ( ) [pure virtual]

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p. 705), and **gazebo::rendering::FPSViewController** (p. 425).

#### 10.227.3.6 virtual void gazebo::rendering::ViewController::Init ( const **math::Vector3** & *_focalPoint* ) [virtual]

Initialize with a focus point.

**Parameters**

| | | |
|---|---|---|
| in | *_focalPoint* | The point to look at. |

Reimplemented in **gazebo::rendering::OrbitViewController** (p. 705).

#### 10.227.3.7 void gazebo::rendering::ViewController::SetEnabled ( bool *_value* )

Set whether the controller is enabled.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | True if the controller is enabled. |

#### 10.227.3.8 virtual void gazebo::rendering::ViewController::Update ( ) [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 186).

Implemented in **gazebo::rendering::OrbitViewController** (p. 706), and **gazebo::rendering::FPSViewController** (p. 425).

### 10.227.4 Member Data Documentation

#### 10.227.4.1 UserCameraPtr gazebo::rendering::ViewController::camera [protected]

Pointer to the camera to control.

#### 10.227.4.2 bool gazebo::rendering::ViewController::enabled [protected]

True if enabled.

#### 10.227.4.3 std::string gazebo::rendering::ViewController::typeString [protected]

Type of view controller.

The documentation for this class was generated from the following file:

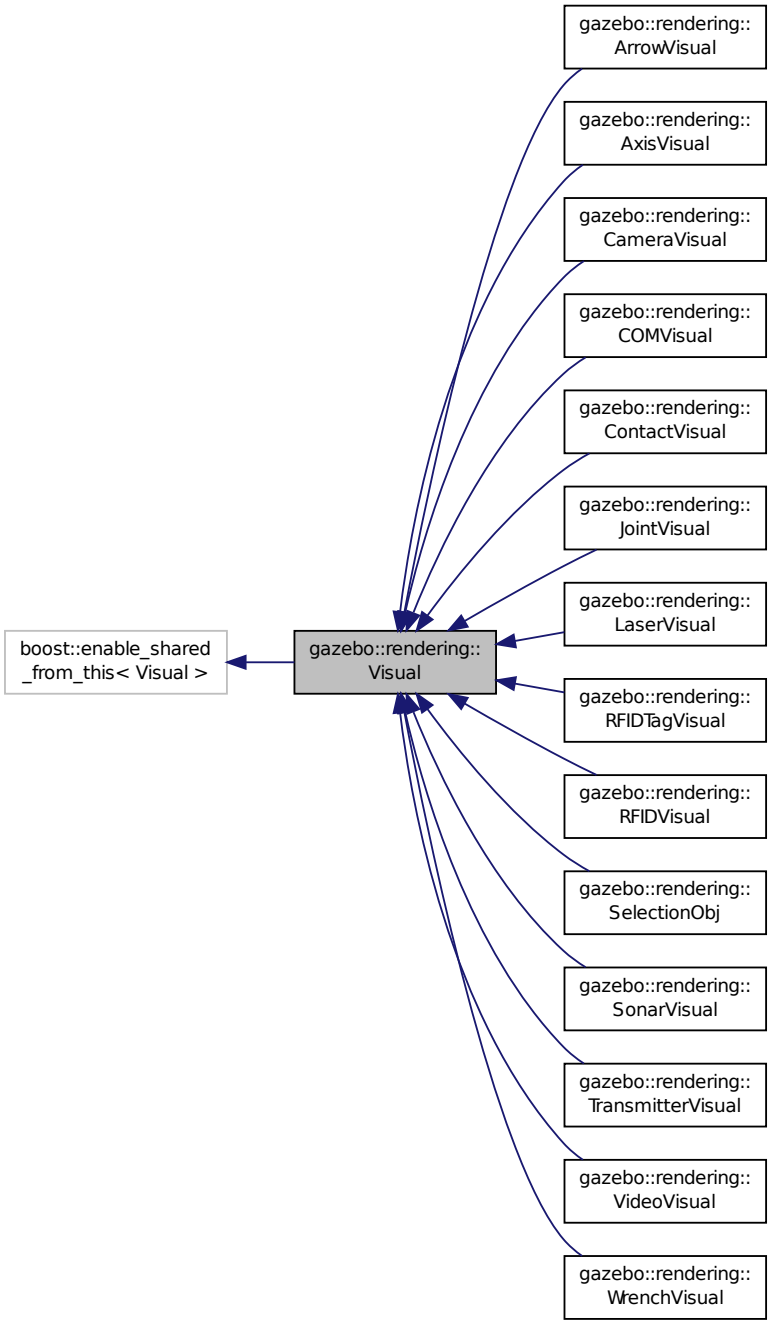
- **ViewController.hh**

## 10.228 gazebo::rendering::Visual Class Reference

A renderable object.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Visual:




## Public Member Functions

- **Visual** (const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)

*Constructor.*

- **Visual** (const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)

  *Constructor.*
- virtual ~**Visual** ()

  *Destructor.*
- void **AttachAxes** ()

  *Attach visualization axes.*
- void **AttachLineVertex** (**DynamicLines** *_line, unsigned int _index)

  *Attach a vertex of a line to the position of the visual.*
- Ogre::MovableObject * **AttachMesh** (const std::string &_meshName, const std::string &_subMesh="", bool _centerSubmesh=false, const std::string &_objName="")

  *Attach a mesh to this visual by name.*
- void **AttachObject** (Ogre::MovableObject *_obj)

  *Attach a renerable object to the visual.*
- void **AttachVisual** (**VisualPtr** _vis)

  *Attach a visual to this visual.*
- void **ClearParent** ()

  *Clear parents.*
- **VisualPtr Clone** (const std::string &_name, **VisualPtr** _newParent)

  *Clone the visual with a new name.*
- **DynamicLines** * **CreateDynamicLine** (**RenderOpType** _type=**RENDERING_LINE_STRIP**)

  *Add a line to the visual.*
- void **DeleteDynamicLine** (**DynamicLines** *_line)

  *Delete a dynamic line.*
- void **DetachObjects** ()

  *Detach all objects.*
- void **DetachVisual** (**VisualPtr** _vis)

  *Detach a visual.*
- void **DetachVisual** (const std::string &_name)

  *Detach a visual.*
- void **DisableTrackVisual** ()

  *Disable tracking of a visual.*
- void **EnableTrackVisual** (**VisualPtr** _vis)

  *Set one visual to track/follow another.*
- void **Fini** ()

  *Helper for the destructor.*
- unsigned int **GetAttachedObjectCount** () const

  *Return the number of attached movable objects.*
- **math::Box GetBoundingBox** () const

  *Get the bounding box for the visual.*
- **VisualPtr GetChild** (unsigned int _index)

  *Get an attached visual based on an index.*
- unsigned int **GetChildCount** ()

  *Get the number of attached visuals.*
- uint32_t **GetId** () const

  *Get the id associated with this visual.*
- std::string **GetMaterialName** () const

*Get the name of the material.*

- std::string **GetMeshName** () const

  *The name of the mesh set in the visual's SDF.*

- std::string **GetName** () const

  *Get the name of the visual.*

- std::string **GetNormalMap** () const

  *Get the normal map.*

- **VisualPtr GetParent** () const

  *Get the parent visual, if one exists.*

- **math::Pose GetPose** () const

  *Get the pose of the visual.*

- **math::Vector3 GetPosition** () const

  *Get the position of the visual.*

- **VisualPtr GetRootVisual** ()

  *Get the root visual.*

- **math::Quaternion GetRotation** () const

  *Get the rotation of the visual.*

- **math::Vector3 GetScale** ()

  *Get the scale.*

- **ScenePtr GetScene** () const

  *Get current.*

- Ogre::SceneNode * **GetSceneNode** () const

  *Return the scene Node of this visual entity.*

- std::string **GetShaderType** () const

  *Get the shader type.*

- std::string **GetSubMeshName** () const

  *Get the name of the sub mesh set in the visual's SDF.*

- float **GetTransparency** ()

  *Get the transparency.*

- uint32_t **GetVisibilityFlags** ()

  *Get visibility flags for this visual and all children.*

- bool **GetVisible** () const

  *Get whether the visual is visible.*

- **math::Pose GetWorldPose** () const

  *Get the global pose of the node.*

- bool **HasAttachedObject** (const std::string &_name)

  *Returns true if an object with _name is attached.*

- void **Init** ()

  *Helper for the contructor.*

- void **InsertMesh** (const std::string &_meshName, const std::string &_subMesh="", bool _centerSubmesh=false)

  *Insert a mesh into **Ogre** (p. 130).*

- bool **IsPlane** () const

  *Return true if the visual is a plane.*

- bool **IsStatic** () const

  *Return true if the visual is a static geometry.*

- void **Load** (sdf::ElementPtr _sdf)

  *Load the visual with a set of parameters.*

- virtual void **Load** ()

  *Load the visual with default parameters.*
- void **LoadFromMsg** (ConstVisualPtr &_msg)

  *Load from a message.*
- void **LoadPlugin** (const std::string &_filename, const std::string &_name, sdf::ElementPtr _sdf)

  *Load a plugin.*
- void **MakeStatic** ()

  *Make the visual objects static renderables.*
- void **MoveToPosition** (const **math::Pose** &_pose, double _time)

  *Move to a pose and over a given time.*
- void **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _onComplete=**NULL**)

  *Move to a series of pose and over a given time.*
- void **RemovePlugin** (const std::string &_name)

  *Remove a running plugin.*
- void **SetAmbient** (const **common::Color** &_color)

  *Set the ambient color of the visual.*
- void **SetCastShadows** (bool _shadows)

  *Set whether the visual should cast shadows.*
- void **SetDiffuse** (const **common::Color** &_color)

  *Set the diffuse color of the visual.*
- virtual void **SetEmissive** (const **common::Color** &_color)

  *Set the emissive value.*
- void **SetHighlighted** (bool _highlighted)

  *Set the visual to be visually highlighted.*
- void **SetId** (uint32_t _id)

  *Set the id associated with this visual.*
- void **SetMaterial** (const std::string &_materialName, bool _unique=true)

  *Set the material.*
- void **SetName** (const std::string &_name)

  *Set the name of the visual.*
- void **SetNormalMap** (const std::string &_nmap)

  *Set the normal map.*
- void **SetPose** (const **math::Pose** &_pose)

  *Set the pose of the visual.*
- void **SetPosition** (const **math::Vector3** &_pos)

  *Set the position of the visual.*
- void **SetRibbonTrail** (bool _value, const **common::Color** &_initialColor, const **common::Color** &_changeColor)

  *True on or off a ribbon trail.*
- void **SetRotation** (const **math::Quaternion** &_rot)

  *Set the rotation of the visual.*
- void **SetScale** (const **math::Vector3** &_scale)

  *Set the scale.*
- void **SetScene** (**ScenePtr** _scene)

  *Set current scene.*
- void **SetShaderType** (const std::string &_type)

*Set the shader type for the visual's material.*

- void **SetSkeletonPose** (const msgs::PoseAnimation &_pose)

  *Set animation skeleton pose.*

- void **SetSpecular** (const **common::Color** &_color)

  *Set the specular color of the visual.*

- void **SetTransparency** (float _trans)

  *Set the transparency.*

- void **SetVisibilityFlags** (uint32_t _flags)

  *Set visibility flags for this visual and all children.*

- void **SetVisible** (bool _visible, bool _cascade=true)

  *Set whether the visual is visible.*

- void **SetWireframe** (bool _show)

  *Enable or disable wireframe for this visual.*

- void **SetWorldPose** (const **math::Pose** _pose)

  *Set the world pose of the visual.*

- void **SetWorldPosition** (const **math::Vector3** &_pos)

  *Set the world linear position of the visual.*

- void **SetWorldRotation** (const **math::Quaternion** &_rot)

  *Set the world orientation of the visual.*

- void **ShowBoundingBox** ()

  *Display the bounding box visual.*

- void **ShowCollision** (bool _show)

  *Display the collision visuals.*

- void **ShowCOM** (bool _show)

  *Display Center of Mass visuals.*

- void **ShowJoints** (bool _show)

  *Display joint visuals.*

- void **ShowSkeleton** (bool _show)

  *Display the skeleton visuals.*

- void **ToggleVisible** ()

  *Toggle whether this visual is visible.*

- void **Update** ()

  *Update the visual.*

- void **UpdateFromMsg** (ConstVisualPtr &_msg)

  *Update a visual based on a message.*

## Static Public Member Functions

- static void **InsertMesh** (const **common::Mesh** *_mesh, const std::string &_subMesh="", bool _centerSubmesh=false)

  *Insert a mesh into **Ogre** (p. 130).*

### Protected Attributes

- **VisualPtr parent**

  *Parent visual.*
- **ScenePtr scene**

  *Pointer to the visual's scene.*
- Ogre::SceneNode * **sceneNode**

  *Pointer to the visual's scene node in* ***Ogre*** *(p. 130).*

## 10.228.1 Detailed Description

A renderable object.

## 10.228.2 Constructor & Destructor Documentation

### 10.228.2.1 gazebo::rendering::Visual::Visual ( const std::string & *_name,* VisualPtr *_parent,* bool *_useRTShader* = `true` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual. |
| in | *_parent* | Parent of the visual. |
| in | *_useRTShader* | True if the visual should use the real-time shader system (RTShader). |

### 10.228.2.2 gazebo::rendering::Visual::Visual ( const std::string & *_name,* ScenePtr *_scene,* bool *_useRTShader* = `true` )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual. |
| in | *_scene* | **Scene** (p. 814) containing the visual. |
| in | *_useRTShader* | True if the visual should use the real-time shader system (RTShader). |

### 10.228.2.3 virtual gazebo::rendering::Visual::~Visual ( ) `[virtual]`

Destructor.

## 10.228.3 Member Function Documentation

### 10.228.3.1 void gazebo::rendering::Visual::AttachAxes ( )

Attach visualization axes.

### 10.228.3.2 void gazebo::rendering::Visual::AttachLineVertex ( DynamicLines * *_line,* unsigned int *_index* )

Attach a vertex of a line to the position of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_line* | Line to attach to this visual. |
| in | *_index* | Index of the line vertex to attach. |

#### 10.228.3.3 Ogre::MovableObject* gazebo::rendering::Visual::AttachMesh ( const std::string & *_meshName*, const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false, const std::string & *_objName* = " " )

Attach a mesh to this visual by name.

**Parameters**

| | | |
|---|---|---|
| in | *_meshName* | Name of the mesh. |
| in | *_subMesh* | Name of the submesh. Empty string to use all submeshes. |
| in | *_centerSubmesh* | True to center a submesh. |
| in | *_objName* | Name of the attached Object to put the mesh onto. |

#### 10.228.3.4 void gazebo::rendering::Visual::AttachObject ( Ogre::MovableObject * *_obj* )

Attach a renerable object to the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_obj* | A movable object to attach to the visual. |

#### 10.228.3.5 void gazebo::rendering::Visual::AttachVisual ( VisualPtr *_vis* )

Attach a visual to this visual.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | **Visual** (p. 1121) to attach. |

#### 10.228.3.6 void gazebo::rendering::Visual::ClearParent ( )

Clear parents.

#### 10.228.3.7 VisualPtr gazebo::rendering::Visual::Clone ( const std::string & *_name*, VisualPtr *_newParent* )

Clone the visual with a new name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the cloned **Visual** (p. 1121). |
| in | *_newParent* | Parent of the cloned **Visual** (p. 1121). |

**Returns**

The visual.

#### 10.228.3.8 DynamicLines* gazebo::rendering::Visual::CreateDynamicLine ( RenderOpType _type = RENDERING_LINE_STRIP )

Add a line to the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | The type of line to make. |

**Returns**

A pointer to the new dynamic line.

#### 10.228.3.9 void gazebo::rendering::Visual::DeleteDynamicLine ( DynamicLines * _line )

Delete a dynamic line.

**Parameters**

| | | |
|---|---|---|
| in | *_line* | Pointer to the line to delete. |

#### 10.228.3.10 void gazebo::rendering::Visual::DetachObjects ( )

Detach all objects.

#### 10.228.3.11 void gazebo::rendering::Visual::DetachVisual ( VisualPtr _vis )

Detach a visual.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | **Visual** (p. 1121) to detach. |

#### 10.228.3.12 void gazebo::rendering::Visual::DetachVisual ( const std::string & _name )

Detach a visual.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual to detach. |

#### 10.228.3.13 void gazebo::rendering::Visual::DisableTrackVisual ( )

Disable tracking of a visual.

#### 10.228.3.14 void gazebo::rendering::Visual::EnableTrackVisual ( VisualPtr _vis )

Set one visual to track/follow another.

**Parameters**

| | | |
|---|---|---|
| in | *_vis* | **Visual** (p. 1121) to track. |

#### 10.228.3.15 void gazebo::rendering::Visual::Fini ( )

Helper for the destructor.

#### 10.228.3.16 unsigned int gazebo::rendering::Visual::GetAttachedObjectCount ( ) const

Return the number of attached movable objects.

**Returns**

The number of attached movable objects.

#### 10.228.3.17 math::Box gazebo::rendering::Visual::GetBoundingBox ( ) const

Get the bounding box for the visual.

**Returns**

The bounding box in world coordinates.

#### 10.228.3.18 VisualPtr gazebo::rendering::Visual::GetChild ( unsigned int *_index* )

Get an attached visual based on an index.

Index should be between 0 and **Visual::GetChildCount** (p. 1130).

**Parameters**

| | | |
|---|---|---|
| in | *_index* | Index of the child to retreive. |

**Returns**

Pointer to the child visual, NULL if index is invalid.

#### 10.228.3.19 unsigned int gazebo::rendering::Visual::GetChildCount ( )

Get the number of attached visuals.

**Returns**

The number of children.

#### 10.228.3.20 uint32_t gazebo::rendering::Visual::GetId ( ) const

Get the id associated with this visual.

#### 10.228.3.21 std::string gazebo::rendering::Visual::GetMaterialName ( ) const

Get the name of the material.

**Returns**

The name of the visual applied to this visual.

#### 10.228.3.22 std::string gazebo::rendering::Visual::GetMeshName ( ) const

The name of the mesh set in the visual's SDF.

**Returns**

Name of the mesh.

#### 10.228.3.23 std::string gazebo::rendering::Visual::GetName ( ) const

Get the name of the visual.

**Returns**

The name of the visual.

#### 10.228.3.24 std::string gazebo::rendering::Visual::GetNormalMap ( ) const

Get the normal map.

**Returns**

The name of the normal map material.

#### 10.228.3.25 VisualPtr gazebo::rendering::Visual::GetParent ( ) const

Get the parent visual, if one exists.

**Returns**

Pointer to the parent visual, NULL if no parent.

#### 10.228.3.26 math::Pose gazebo::rendering::Visual::GetPose ( ) const

Get the pose of the visual.

**Returns**

The **Visual** (p. 1121)'s pose.

#### 10.228.3.27 math::Vector3 gazebo::rendering::Visual::GetPosition ( ) const

Get the position of the visual.

**Returns**

The visual's position.

#### 10.228.3.28 VisualPtr gazebo::rendering::Visual::GetRootVisual ( )

Get the root visual.

**Returns**

The root visual, which is one level below the world visual.

#### 10.228.3.29 math::Quaternion gazebo::rendering::Visual::GetRotation ( ) const

Get the rotation of the visual.

**Returns**

The visual's rotation.

#### 10.228.3.30 math::Vector3 gazebo::rendering::Visual::GetScale ( )

Get the scale.

**Returns**

The scaling factor.

#### 10.228.3.31 ScenePtr gazebo::rendering::Visual::GetScene ( ) const

Get current.

**Returns**

Pointer to the scene.

#### 10.228.3.32 Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode ( ) const

Return the scene Node of this visual entity.

**Returns**

The **Ogre** (p. 130) scene node.

#### 10.228.3.33 std::string gazebo::rendering::Visual::GetShaderType ( ) const

Get the shader type.

**Returns**

String of the shader type: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".

#### 10.228.3.34 std::string gazebo::rendering::Visual::GetSubMeshName ( ) const

Get the name of the sub mesh set in the visual's SDF.

**Returns**

Name of the submesh. Empty string if no submesh is specified.

#### 10.228.3.35 float gazebo::rendering::Visual::GetTransparency ( )

Get the transparency.

**Returns**

The transparency.

#### 10.228.3.36 uint32_t gazebo::rendering::Visual::GetVisibilityFlags ( )

Get visibility flags for this visual and all children.

**Returns**

The visiblity flags.

**See Also**

**GZ_VISIBILITY_ALL** (p. 1333)
**GZ_VISIBILITY_GUI** (p. 1334)
**GZ_VISIBILITY_SELECTABLE** (p. 1334)

#### 10.228.3.37 bool gazebo::rendering::Visual::GetVisible ( ) const

Get whether the visual is visible.

**Returns**

True if the visual is visible.

#### 10.228.3.38 math::Pose gazebo::rendering::Visual::GetWorldPose ( ) const

Get the global pose of the node.

**Returns**

The pose in the world coordinate frame.

#### 10.228.3.39 bool gazebo::rendering::Visual::HasAttachedObject ( const std::string & *_name* )

Returns true if an object with _name is attached.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of an object to find. |

#### 10.228.3.40 void gazebo::rendering::Visual::Init ( )

Helper for the contructor.

#### 10.228.3.41 void gazebo::rendering::Visual::InsertMesh ( const std::string & *_meshName*, const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false )

Insert a mesh into **Ogre** (p. 130).

**Parameters**

| | | |
|---|---|---|
| in | *_meshName* | Name of the mesh to insert. |
| in | *_subMesh* | Name of the mesh within _meshName to insert. |
| in | *_centerSubmesh* | True to center the submesh. |

#### 10.228.3.42 static void gazebo::rendering::Visual::InsertMesh ( const common::Mesh * *_mesh*, const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false ) [static]

Insert a mesh into **Ogre** (p. 130).

**Parameters**

| | | |
|---|---|---|
| in | *_mesh* | Pointer to the mesh to insert. |
| in | *_subMesh* | Name of the mesh within _meshName to insert. |
| in | *_centerSubmesh* | True to center the submesh. |

#### 10.228.3.43 bool gazebo::rendering::Visual::IsPlane ( ) const

Return true if the visual is a plane.

**Returns**

True if a plane.

#### 10.228.3.44 bool gazebo::rendering::Visual::IsStatic ( ) const

Return true if the visual is a static geometry.

**Returns**

True if the visual is static.

#### 10.228.3.45 void gazebo::rendering::Visual::Load ( sdf::ElementPtr _sdf )

Load the visual with a set of parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | Load from an SDF element. |

#### 10.228.3.46 virtual void gazebo::rendering::Visual::Load ( ) [virtual]

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::SelectionObj** (p. 85), **gazebo::rendering::ArrowVisual** (p. 151), **gazebo::rendering::SonarVisual** (p. 983), **gazebo::rendering::TransmitterVisual** (p. 1063), and **gazebo::rendering::AxisVisual** (p. 156).

#### 10.228.3.47 void gazebo::rendering::Visual::LoadFromMsg ( ConstVisualPtr & _msg )

Load from a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | A visual message. |

#### 10.228.3.48 void gazebo::rendering::Visual::LoadPlugin ( const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf )

Load a plugin.

**Parameters**

| | |
|---|---|
| *_filename* | The filename of the plugin |
| *_name* | A unique name for the plugin |
| *_sdf* | The SDF to pass into the plugin. |

#### 10.228.3.49 void gazebo::rendering::Visual::MakeStatic ( )

Make the visual objects static renderables.

#### 10.228.3.50 void gazebo::rendering::Visual::MoveToPosition ( const math::Pose & *_pose*, double *_time* )

Move to a pose and over a given time.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | Pose the visual will end at. |
| in | *_time* | Time it takes the visual to move to the pose. |

#### 10.228.3.51 void gazebo::rendering::Visual::MoveToPositions ( const std::vector< math::Pose > & *_pts*, double *_time*, boost::function< void() > *_onComplete* = NULL )

Move to a series of pose and over a given time.

**Parameters**

| | | |
|---|---|---|
| in | *_poses* | Series of poses the visual will move to. |
| in | *_time* | Time it takes the visual to move to the pose. |
| in | *_onComplete* | Callback used when the move is complete. |

#### 10.228.3.52 void gazebo::rendering::Visual::RemovePlugin ( const std::string & *_name* )

Remove a running plugin.

**Parameters**

| | |
|---|---|
| *_name* | The unique name of the plugin to remove |

#### 10.228.3.53 void gazebo::rendering::Visual::SetAmbient ( const common::Color & *_color* )

Set the ambient color of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | The ambient color. |

#### 10.228.3.54 void gazebo::rendering::Visual::SetCastShadows ( bool *_shadows* )

Set whether the visual should cast shadows.

**Parameters**

| | | |
|---|---|---|
| in | *_shadows* | True to enable shadows. |

#### 10.228.3.55 void gazebo::rendering::Visual::SetDiffuse ( const common::Color & _color )

Set the diffuse color of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | Set the diffuse color. |

#### 10.228.3.56 virtual void gazebo::rendering::Visual::SetEmissive ( const common::Color & _color ) [virtual]

Set the emissive value.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | The emissive color. |

Reimplemented in **gazebo::rendering::LaserVisual** (p. 535).

#### 10.228.3.57 void gazebo::rendering::Visual::SetHighlighted ( bool _highlighted )

Set the visual to be visually highlighted.

This is most often used when an object is selected by a user via the GUI.

**Parameters**

| | | |
|---|---|---|
| in | *_highlighted* | True to enable the highlighting. |

#### 10.228.3.58 void gazebo::rendering::Visual::SetId ( uint32_t _id )

Set the id associated with this visual.

#### 10.228.3.59 void gazebo::rendering::Visual::SetMaterial ( const std::string & _materialName, bool _unique = true )

Set the material.

**Parameters**

| | | |
|---|---|---|
| in | *_materialName* | The name of the material. |
| in | *_unique* | True to make the material unique, which allows the material to change without changing materials that originally had the same name. |

#### 10.228.3.60 void gazebo::rendering::Visual::SetName ( const std::string & _name )

Set the name of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual |

#### 10.228.3.61 void gazebo::rendering::Visual::SetNormalMap ( const std::string & _nmap )

Set the normal map.

**Parameters**

| | | |
|---|---|---|
| in | *_nmap* | Name of the normal map material. |

#### 10.228.3.62 void gazebo::rendering::Visual::SetPose ( const math::Pose & _pose )

Set the pose of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | The new pose of the visual. |

#### 10.228.3.63 void gazebo::rendering::Visual::SetPosition ( const math::Vector3 & _pos )

Set the position of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_pos* | The position to set the visual to. |

#### 10.228.3.64 void gazebo::rendering::Visual::SetRibbonTrail ( bool _value, const common::Color & _initialColor, const common::Color & _changeColor )

True on or off a ribbon trail.

**Parameters**

| | | |
|---|---|---|
| in | *_value* | True to enable ribbon trail. |
| in | *_initialColor* | The initial color of the ribbon trail. |
| in | *_changeColor* | Color to change too as the trail grows. |

#### 10.228.3.65 void gazebo::rendering::Visual::SetRotation ( const math::Quaternion & _rot )

Set the rotation of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_rot* | The rotation of the visual. |

#### 10.228.3.66 void gazebo::rendering::Visual::SetScale ( const math::Vector3 & _scale )

Set the scale.

**Parameters**

| | | |
|---|---|---|
| in | *_scale* | The scaling factor for the visual. |

#### 10.228.3.67 void gazebo::rendering::Visual::SetScene ( ScenePtr *_scene* )

Set current scene.

**Parameters**

| | | |
|---|---|---|
| in | *_scene* | Pointer to the scene. |

#### 10.228.3.68 void gazebo::rendering::Visual::SetShaderType ( const std::string & *_type* )

Set the shader type for the visual's material.

**Parameters**

| | | |
|---|---|---|
| in | *_type* | Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space". |

#### 10.228.3.69 void gazebo::rendering::Visual::SetSkeletonPose ( const msgs::PoseAnimation & *_pose* )

Set animation skeleton pose.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | Skelton message |

#### 10.228.3.70 void gazebo::rendering::Visual::SetSpecular ( const common::Color & *_color* )

Set the specular color of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_color* | Specular color. |

#### 10.228.3.71 void gazebo::rendering::Visual::SetTransparency ( float *_trans* )

Set the transparency.

**Parameters**

| | | |
|---|---|---|
| in | *_trans* | The transparency, between 0 and 1 where 0 is no transparency. |

#### 10.228.3.72 void gazebo::rendering::Visual::SetVisibilityFlags ( uint32_t *_flags* )

Set visibility flags for this visual and all children.

**Parameters**

| | | |
|---|---|---|
| in | *_flags* | The visiblity flags. |

**See Also**

**GZ_VISIBILITY_ALL** (p. 1333)
**GZ_VISIBILITY_GUI** (p. 1334)
**GZ_VISIBILITY_SELECTABLE** (p. 1334)

#### 10.228.3.73 void gazebo::rendering::Visual::SetVisible ( bool *_visible*, bool *_cascade* = `true` )

Set whether the visual is visible.

**Parameters**

| | | |
|---|---|---|
| in | *_visible* | set this node visible. |
| in | *_cascade* | setting this parameter in children too. |

#### 10.228.3.74 void gazebo::rendering::Visual::SetWireframe ( bool *_show* )

Enable or disable wireframe for this visual.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to enable wireframe for this visual. |

#### 10.228.3.75 void gazebo::rendering::Visual::SetWorldPose ( const math::Pose *_pose* )

Set the world pose of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | Pose of the visual in the world coordinate frame. |

#### 10.228.3.76 void gazebo::rendering::Visual::SetWorldPosition ( const math::Vector3 & *_pos* )

Set the world linear position of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_pose* | Position in the world coordinate frame. |

#### 10.228.3.77 void gazebo::rendering::Visual::SetWorldRotation ( const math::Quaternion & *_rot* )

Set the world orientation of the visual.

**Parameters**

| | | |
|---|---|---|
| in | *_rot* | Rotation in the world coordinate frame. |

#### 10.228.3.78 void gazebo::rendering::Visual::ShowBoundingBox ( )

Display the bounding box visual.

#### 10.228.3.79 void gazebo::rendering::Visual::ShowCollision ( bool *_show* )

Display the collision visuals.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to show visuals labeled as collision objects. |

#### 10.228.3.80 void gazebo::rendering::Visual::ShowCOM ( bool *_show* )

Display Center of Mass visuals.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to show center of mass visualizations. |

#### 10.228.3.81 void gazebo::rendering::Visual::ShowJoints ( bool *_show* )

Display joint visuals.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to show joint visualizations. |

#### 10.228.3.82 void gazebo::rendering::Visual::ShowSkeleton ( bool *_show* )

Display the skeleton visuals.

**Parameters**

| | | |
|---|---|---|
| in | *_show* | True to show skeleton visuals. |

#### 10.228.3.83 void gazebo::rendering::Visual::ToggleVisible ( )

Toggle whether this visual is visible.

#### 10.228.3.84 void gazebo::rendering::Visual::Update ( )

Update the visual.

#### 10.228.3.85 void gazebo::rendering::Visual::UpdateFromMsg ( ConstVisualPtr & *_msg* )

Update a visual based on a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | The visual message. |

### 10.228.4 Member Data Documentation

#### 10.228.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

#### 10.228.4.2 ScenePtr gazebo::rendering::Visual::scene [protected]

Pointer to the visual's scene.

#### 10.228.4.3 Ogre::SceneNode* gazebo::rendering::Visual::sceneNode [protected]

Pointer to the visual's scene node in **Ogre** (p. 130).

The documentation for this class was generated from the following file:

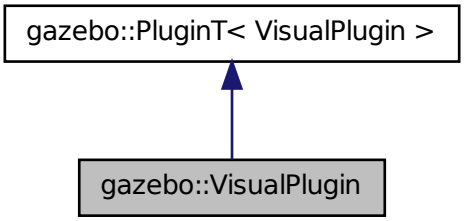
- **Visual.hh**

## 10.229 gazebo::VisualPlugin Class Reference

A plugin loaded within the gzserver on startup.

`#include <Plugin.hh>`

Inheritance diagram for gazebo::VisualPlugin:



### Public Member Functions

- **VisualPlugin** ()
- virtual void **Init** ()

  *Initialize the plugin.*
- virtual void **Load** (**rendering::VisualPtr** _visual, sdf::ElementPtr _sdf)=0

  *Load function.*
- virtual void **Reset** ()

  *Override this method for custom plugin reset behavior.*

### Additional Inherited Members

## 10.229.1 Detailed Description

A plugin loaded within the gzserver on startup.

See `reference`.

## 10.229.2 Constructor & Destructor Documentation

#### 10.229.2.1 gazebo::VisualPlugin::VisualPlugin ( ) [inline]

References gazebo::PluginT< VisualPlugin >::type, and gazebo::VISUAL_PLUGIN.

## 10.229.3 Member Function Documentation

#### 10.229.3.1 virtual void gazebo::VisualPlugin::Init ( ) [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

#### 10.229.3.2 virtual void gazebo::VisualPlugin::Load ( **rendering::VisualPtr** *_visual,* **sdf::ElementPtr** *_sdf* ) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

**Parameters**

| | | |
|---|---|---|
| in | *_visual* | Pointer the Visual Object. |
| in | *_sdf* | Pointer the the SDF element of the plugin. |

#### 10.229.3.3 virtual void gazebo::VisualPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

- **Plugin.hh**

## 10.230 gazebo::rendering::WindowManager Class Reference

Class to mangage render windows.

`#include <rendering/rendering.hh>`

### Public Member Functions

- **WindowManager** ()

  *Constructor.*
- virtual ∼**WindowManager** ()

  *Destructor.*
- int **CreateWindow** (const std::string &_ogreHandle, uint32_t _width, uint32_t _height)

  *Create a window.*
- void **Fini** ()

  *Shutdown all the windows.*
- float **GetAvgFPS** (uint32_t _id)

  *Get the average FPS.*
- uint32_t **GetTriangleCount** (uint32_t _id)

  *Get the triangle count.*
- Ogre::RenderWindow ∗ **GetWindow** (uint32_t _id)

  *Get the render window associated with the given id.*
- void **Moved** (uint32_t _id)

  *Tells **Ogre** (p. 130) the window has moved, and needs updating.*
- void **Resize** (uint32_t _id, int _width, int _height)

  *Resize a window.*
- void **SetCamera** (int _windowId, **CameraPtr** _camera)

  *Attach a camera to a window.*

### 10.230.1 Detailed Description

Class to mangage render windows.

### 10.230.2 Constructor & Destructor Documentation

#### 10.230.2.1 gazebo::rendering::WindowManager::WindowManager ( )

Constructor.

#### 10.230.2.2 virtual gazebo::rendering::WindowManager::∼WindowManager ( ) [virtual]

Destructor.

### 10.230.3 Member Function Documentation

#### 10.230.3.1 int gazebo::rendering::WindowManager::CreateWindow ( const std::string & *_ogreHandle*, uint32_t *_width*, uint32_t *_height* )

Create a window.

**Parameters**

| | | |
|---|---|---|
| in | *_ogreHandle* | String representing the ogre window handle. |
| in | *_width* | With of the window in pixels. |
| in | *_height* | Height of the window in pixels. |

#### 10.230.3.2 void gazebo::rendering::WindowManager::Fini ( )

Shutdown all the windows.

#### 10.230.3.3 float gazebo::rendering::WindowManager::GetAvgFPS ( uint32_t *_id* )

Get the average FPS.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the window. |

**Returns**

The frames per second.

#### 10.230.3.4 uint32_t gazebo::rendering::WindowManager::GetTriangleCount ( uint32_t *_id* )

Get the triangle count.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the window. |

**Returns**

The triangle count.

#### 10.230.3.5 Ogre::RenderWindow* gazebo::rendering::WindowManager::GetWindow ( uint32_t *_id* )

Get the render window associated with the given id.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the window. |

**Returns**

Pointer to the render window, NULL if the id is invalid.

#### 10.230.3.6 void gazebo::rendering::WindowManager::Moved ( uint32_t *_id* )

Tells **Ogre** (p. 130) the window has moved, and needs updating.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | ID of the window. |

#### 10.230.3.7 void gazebo::rendering::WindowManager::Resize ( uint32_t *_id,* int *_width,* int *_height* )

Resize a window.

**Parameters**

| | | |
|---|---|---|
| in | *_id* | Id of the window to resize. |
| in | *_width* | New width of the window. |
| in | *_height* | New height of the window. |

#### 10.230.3.8 void gazebo::rendering::WindowManager::SetCamera ( int *_windowId,* CameraPtr *_camera* )

Attach a camera to a window.

**Parameters**

| | | |
|---|---|---|
| in | *_windowId* | Id of the window to add the camera to. |
| in | *_camera* | Pointer to the camera to attach. |

The documentation for this class was generated from the following file:

- **WindowManager.hh**

## 10.231 gazebo::rendering::WireBox Class Reference

Draws a wireframe box.

`#include <rendering/rendering.hh>`

### Public Member Functions

- **WireBox** (**VisualPtr** _parent, const **math::Box** &_box)

  *Constructor.*
- ~**WireBox** ()

  *Destructor.*
- void **Init** (const **math::Box** &_box)

*Builds the wireframe line list.*

- void **SetVisible** (bool _visible)

  *Set the visibility of the box.*

### 10.231.1 Detailed Description

Draws a wireframe box.

### 10.231.2 Constructor & Destructor Documentation

#### 10.231.2.1 gazebo::rendering::WireBox::WireBox ( VisualPtr *_parent,* const math::Box & *_box* ) [explicit]

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_box* | Dimenision of the box to draw. |

#### 10.231.2.2 gazebo::rendering::WireBox::~WireBox ( )

Destructor.

### 10.231.3 Member Function Documentation

#### 10.231.3.1 void gazebo::rendering::WireBox::Init ( const math::Box & *_box* )

Builds the wireframe line list.

**Parameters**

| | | |
|---|---|---|
| in | *_box* | Box to build a wireframe from. |

#### 10.231.3.2 void gazebo::rendering::WireBox::SetVisible ( bool *_visible* )

Set the visibility of the box.

**Parameters**

| | | |
|---|---|---|
| in | *_visible* | True to make the box visible, False to hide. |

The documentation for this class was generated from the following file:

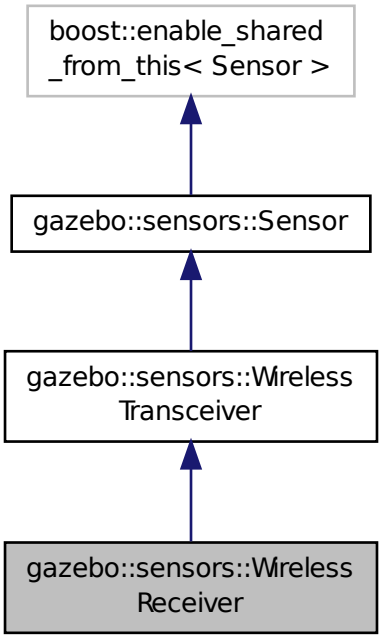
- **WireBox.hh**

## 10.232 gazebo::sensors::WirelessReceiver Class Reference

**Sensor** (p. 837) class for receiving wireless signals.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::WirelessReceiver:




## Public Member Functions

- **WirelessReceiver** ()

  *Constructor.*
- virtual ~**WirelessReceiver** ()

  *Constructor.*
- virtual void **Fini** ()

  *Finalize the sensor.*
- double **GetMaxFreqFiltered** () const

  *Returns the maximum frequency filtered (MHz).*
- double **GetMinFreqFiltered** () const

  *Returns the minimum frequency filtered (MHz).*
- double **GetSensitivity** () const

  *Returns the receiver sensitivity (dBm).*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

## Additional Inherited Members

### 10.232.1 Detailed Description

**Sensor** (p. 837) class for receiving wireless signals.

### 10.232.2 Constructor & Destructor Documentation

#### 10.232.2.1 gazebo::sensors::WirelessReceiver::WirelessReceiver ( )

Constructor.

#### 10.232.2.2 virtual gazebo::sensors::WirelessReceiver::~WirelessReceiver ( ) [virtual]

Constructor.

### 10.232.3 Member Function Documentation

#### 10.232.3.1 virtual void gazebo::sensors::WirelessReceiver::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1152).

#### 10.232.3.2 double gazebo::sensors::WirelessReceiver::GetMaxFreqFiltered ( ) const

Returns the maximum frequency filtered (MHz).

**Returns**

Reception frequency (MHz).

#### 10.232.3.3 double gazebo::sensors::WirelessReceiver::GetMinFreqFiltered ( ) const

Returns the minimum frequency filtered (MHz).

**Returns**

Reception frequency (MHz).

#### 10.232.3.4 double gazebo::sensors::WirelessReceiver::GetSensitivity ( ) const

Returns the receiver sensitivity (dBm).

**Returns**

Receiver sensitivity (dBm).

#### 10.232.3.5 virtual void gazebo::sensors::WirelessReceiver::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1152).

#### 10.232.3.6 virtual void gazebo::sensors::WirelessReceiver::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1152).

The documentation for this class was generated from the following file:

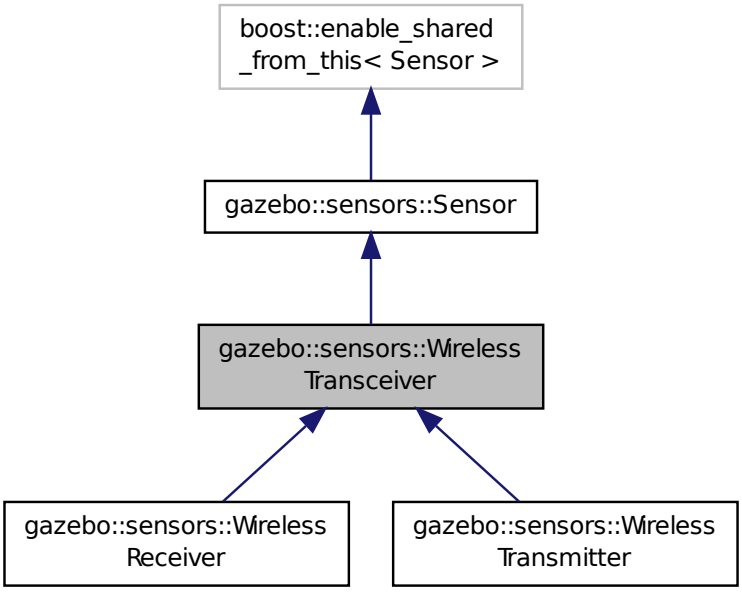
- **WirelessReceiver.hh**

## 10.233 gazebo::sensors::WirelessTransceiver Class Reference

**Sensor** (p. 837) class for receiving wireless signals.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::WirelessTransceiver:



### Public Member Functions

- **WirelessTransceiver** ()

  *Constructor.*
- ∼**WirelessTransceiver** ()

  *Constructor.*
- virtual void **Fini** ()

  *Finalize the sensor.*
- double **GetGain** () const

  *Returns the antenna's gain of the receiver (dBi).*
- double **GetPower** () const

  *Returns the receiver power (dBm).*
- virtual std::string **GetTopic** () const

  *Returns the topic name as set in SDF.*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

### Protected Attributes

- double **gain**

  *Antenna's gain of the receiver (dBi).*
- boost::weak_ptr< **physics::Link** > **parentEntity**

  *Parent entity which the sensor is attached to.*
- double **power**

  *Receiver's power (dBm).*
- **transport::PublisherPtr pub**

  *Publisher to publish propagation model data.*
- **math::Pose referencePose**

  ***Sensor*** *(*p. *837) reference pose.*

### Additional Inherited Members

## 10.233.1 Detailed Description

**Sensor** (p. 837) class for receiving wireless signals.

## 10.233.2 Constructor & Destructor Documentation

#### 10.233.2.1 gazebo::sensors::WirelessTransceiver::WirelessTransceiver ( )

Constructor.

#### 10.233.2.2 gazebo::sensors::WirelessTransceiver::∼WirelessTransceiver ( )

Constructor.

### 10.233.3 Member Function Documentation

#### 10.233.3.1 virtual void gazebo::sensors::WirelessTransceiver::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 841).

Reimplemented in **gazebo::sensors::WirelessReceiver** (p. 1149).

#### 10.233.3.2 double gazebo::sensors::WirelessTransceiver::GetGain ( ) const

Returns the antenna's gain of the receiver (dBi).

**Returns**

Antenna's gain of the receiver (dBi).

#### 10.233.3.3 double gazebo::sensors::WirelessTransceiver::GetPower ( ) const

Returns the receiver power (dBm).

**Returns**

Receiver power (dBm).

#### 10.233.3.4 virtual std::string gazebo::sensors::WirelessTransceiver::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

**Returns**

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 843).

#### 10.233.3.5 virtual void gazebo::sensors::WirelessTransceiver::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 844).

Reimplemented in **gazebo::sensors::WirelessTransmitter** (p. 1156), and **gazebo::sensors::WirelessReceiver** (p. 1150).

#### 10.233.3.6 virtual void gazebo::sensors::WirelessTransceiver::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::Sensor** (p. 845).

Reimplemented in **gazebo::sensors::WirelessTransmitter** (p. 1156), and **gazebo::sensors::WirelessReceiver** (p. 1150).

### 10.233.4 Member Data Documentation

#### 10.233.4.1 double gazebo::sensors::WirelessTransceiver::gain [protected]

Antenna's gain of the receiver (dBi).

#### 10.233.4.2 boost::weak_ptr<physics::Link> gazebo::sensors::WirelessTransceiver::parentEntity [protected]

Parent entity which the sensor is attached to.

#### 10.233.4.3 double gazebo::sensors::WirelessTransceiver::power [protected]

Receiver's power (dBm).

#### 10.233.4.4 transport::PublisherPtr gazebo::sensors::WirelessTransceiver::pub [protected]

Publisher to publish propagation model data.

#### 10.233.4.5 math::Pose gazebo::sensors::WirelessTransceiver::referencePose [protected]

**Sensor** (p. 837) reference pose.

The documentation for this class was generated from the following file:

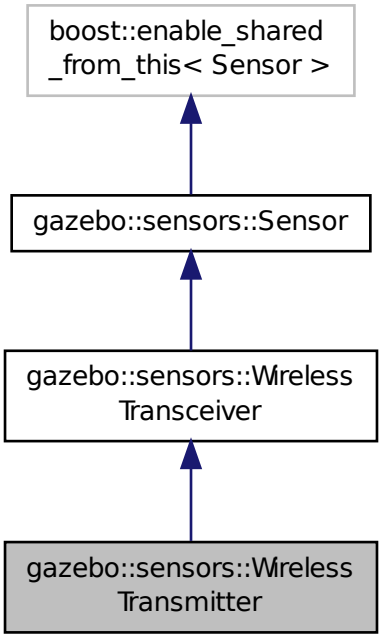
- **WirelessTransceiver.hh**

## 10.234 gazebo::sensors::WirelessTransmitter Class Reference

Transmitter to send wireless signals.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::WirelessTransmitter:




## Public Member Functions

- **WirelessTransmitter** ()

  *Constructor.*
- virtual ~**WirelessTransmitter** ()

  *Destructor.*
- std::string **GetESSID** () const

  *Returns the Service Set Identifier (network name).*
- double **GetFreq** () const

  *Returns reception frequency (MHz).*
- double **GetSignalStrength** (const **math::Pose** &_receiver, const double rxGain)

  *Returns the signal strength in a given world's point (dBm).*
- virtual void **Init** ()

  *Initialize the sensor.*
- virtual void **Load** (const std::string &_worldName)

  *Load the sensor with default parameters.*

## Static Public Attributes

- static const double **ModelStdDesv**

*Std desv of the Gaussian random variable used in the propagation model.*

- static const double **NEmpty**

  *Constant used in the propagation model when there are no obstacles between transmitter and receiver.*

- static const double **NObstacle**

  *Constant used in the propagation model when there are obstacles between transmitter and receiver.*

### Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  *This gets overwritten by derived sensor types.*

### Protected Attributes

- double **freq**

  *Reception frequency (MHz).*

## 10.234.1 Detailed Description

Transmitter to send wireless signals.

## 10.234.2 Constructor & Destructor Documentation

#### 10.234.2.1 gazebo::sensors::WirelessTransmitter::WirelessTransmitter ( )

Constructor.

#### 10.234.2.2 virtual gazebo::sensors::WirelessTransmitter::~WirelessTransmitter ( ) [virtual]

Destructor.

## 10.234.3 Member Function Documentation

#### 10.234.3.1 std::string gazebo::sensors::WirelessTransmitter::GetESSID ( ) const

Returns the Service Set Identifier (network name).

**Returns**

Service Set Identifier (network name).

#### 10.234.3.2 double gazebo::sensors::WirelessTransmitter::GetFreq ( ) const

Returns reception frequency (MHz).

**Returns**

Reception frequency (MHz).

#### 10.234.3.3 double gazebo::sensors::WirelessTransmitter::GetSignalStrength ( const math::Pose & *_receiver,* const double *rxGain* )

Returns the signal strength in a given world's point (dBm).

**Returns**

Signal strength in a world's point (dBm).

#### 10.234.3.4 virtual void gazebo::sensors::WirelessTransmitter::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1152).

#### 10.234.3.5 virtual void gazebo::sensors::WirelessTransmitter::Load ( const std::string & *_worldName* ) [virtual]

Load the sensor with default parameters.

**Parameters**

| | | |
|---|---|---|
| in | *_worldName* | Name of world to load from. |

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1152).

#### 10.234.3.6 virtual void gazebo::sensors::WirelessTransmitter::UpdateImpl ( bool ) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

**Parameters**

| | | |
|---|---|---|
| in | *_force* | True if update is forced, false if not |

Reimplemented from **gazebo::sensors::Sensor** (p. 846).

### 10.234.4 Member Data Documentation

#### 10.234.4.1 double gazebo::sensors::WirelessTransmitter::freq [protected]

Reception frequency (MHz).

#### 10.234.4.2 const double gazebo::sensors::WirelessTransmitter::ModelStdDesv [static]

Std desv of the Gaussian random variable used in the propagation model.

#### 10.234.4.3 const double gazebo::sensors::WirelessTransmitter::NEmpty [static]

Constant used in the propagation model when there are no obstacles between transmitter and receiver.

#### 10.234.4.4 const double gazebo::sensors::WirelessTransmitter::NObstacle [static]

Constant used in the propagation model when there are obstacles between transmitter and receiver.

The documentation for this class was generated from the following file:

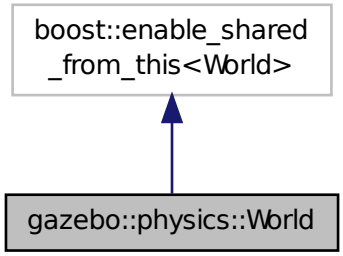
- **WirelessTransmitter.hh**

## 10.235 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::World:




### Public Member Functions

- **World** (const std::string &_name="")
  *Constructor.*
- ~**World** ()
  *Destructor.*
- void **Clear** ()
  *Remove all entities from the world.*
- void **ClearModels** ()
  *Remove all entities from the world.*
- void **DisableAllModels** ()
  *Disable all links in all the models.*
- void **EnableAllModels** ()
  *Enable all links in all the models.*

- void **EnablePhysicsEngine** (bool _enable)

  *enable/disable physics engine during World::Update.*
- void **Fini** ()

  *Finalize the world.*
- **BasePtr GetByName** (const std::string &_name)

  *Get an element by name.*
- bool **GetEnablePhysicsEngine** ()

  *check if physics engine is enabled/disabled.*
- **EntityPtr GetEntity** (const std::string &_name)

  *Get a pointer to an **Entity** (p. 379) based on a name.*
- **EntityPtr GetEntityBelowPoint** (const **math::Vector3** &_pt)

  *Get the nearest entity below a point.*
- **ModelPtr GetModel** (unsigned int _index) const

  *Get a model based on an index.*
- **ModelPtr GetModel** (const std::string &_name)

  *Get a model by name.*
- **ModelPtr GetModelBelowPoint** (const **math::Vector3** &_pt)

  *Get the nearest model below a point.*
- unsigned int **GetModelCount** () const

  *Get the number of models.*
- **Model_V GetModels** () const

  *Get a list of all the models.*
- std::string **GetName** () const

  *Get the name of the world.*
- **common::Time GetPauseTime** () const

  *Get the amount of time simulation has been paused.*
- **PhysicsEnginePtr GetPhysicsEngine** () const

  *Return the physics engine.*
- **common::Time GetRealTime** () const

  *Get the real time (elapsed time).*
- bool **GetRunning** () const

  *Return the running state of the world.*
- **EntityPtr GetSelectedEntity** () const

  *Get the selected **Entity** (p. 379).*
- boost::mutex ∗ **GetSetWorldPoseMutex** () const

  *Get the set world pose mutex.*
- **common::Time GetSimTime** () const

  *Get the world simulation time, note if you want the PC wall clock call **common::Time::GetWallTime** (p. 1037).*
- **common::SphericalCoordinatesPtr GetSphericalCoordinates** () const

  *Return the spherical coordinates converter.*
- **common::Time GetStartTime** () const

  *Get the wall time simulation was started.*
- void **Init** ()

  *Initialize the world.*
- void **InsertModelFile** (const std::string &_sdfFilename)

  *Insert a model from an SDF file.*
- void **InsertModelSDF** (const sdf::SDF &_sdf)

*Insert a model using SDF.*

- void **InsertModelString** (const std::string &_sdfString)

  *Insert a model from an SDF string.*
- bool **IsLoaded** () const

  *Return true if the world has been loaded.*
- bool **IsPaused** () const

  *Returns the state of the simulation true if paused.*
- void **Load** (sdf::ElementPtr _sdf)

  *Load the world using SDF parameters.*
- void **LoadPlugin** (const std::string &_filename, const std::string &_name, sdf::ElementPtr _sdf)

  *Load a plugin.*
- void **PrintEntityTree** ()

  *Print **Entity** (p. 379) tree.*
- void **PublishModelPose** (**physics::ModelPtr** _model)

  *Publish pose updates for a model.*
- void **RemovePlugin** (const std::string &_name)

  *Remove a running plugin.*
- void **Reset** ()

  *Reset time and model poses, configurations in simulation.*
- void **ResetEntities** (**Base::EntityType** _type=**Base::BASE**)

  *Reset with options.*
- void **ResetTime** ()

  *Reset simulation time back to zero.*
- void **Run** (unsigned int _iterations=0)

  *Run the world in a thread.*
- void **Save** (const std::string &_filename)

  *Save a world to a file.*
- void **SetPaused** (bool _p)

  *Set whether the simulation is paused.*
- void **SetSimTime** (const **common::Time** &_t)

  *Set the sim time.*
- void **SetState** (const **WorldState** &_state)

  *Set the current world state.*
- void **StepWorld** (int _steps)

  *Step callback.*
- void **Stop** ()

  *Stop the world.*
- std::string **StripWorldName** (const std::string &_name) const

  *Return a version of the name with "<world_name>::" removed.*
- void **UpdateStateSDF** ()

  *Update the state SDF value from the current state.*

## Public Attributes

- std::list< **Entity** * > **dirtyPoses**

  *when physics engine makes an update and changes a link pose, this flag is set to trigger **Entity::SetWorldPose** (p. 388) on the **physics::Link** (p. 542) in World::Update.*

### 10.235.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 1157) is the container for all models and their components (links, joints, sensors, plugins, etc), and **WorldPlugin** (p. 1169) instances. Many core function are also handled in the **World** (p. 1157), including physics update, model updates, and message processing.

### 10.235.2 Constructor & Destructor Documentation

#### 10.235.2.1 gazebo::physics::World::World ( const std::string & *_name* = " " ) [explicit]

Constructor.

Constructor for the **World** (p. 1157). Must specify a unique name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the world. |

#### 10.235.2.2 gazebo::physics::World::~World ( )

Destructor.

### 10.235.3 Member Function Documentation

#### 10.235.3.1 void gazebo::physics::World::Clear ( )

Remove all entities from the world.

This function has delayed effect. Models are cleared at the end of the current update iteration.

#### 10.235.3.2 void gazebo::physics::World::ClearModels ( )

Remove all entities from the world.

Implementation of **World::Clear** (p. 1160)

#### 10.235.3.3 void gazebo::physics::World::DisableAllModels ( )

Disable all links in all the models.

Disable is a physics concept. Disabling means that the physics engine should not update an entity.

#### 10.235.3.4 void gazebo::physics::World::EnableAllModels ( )

Enable all links in all the models.

Enable is a physics concept. Enabling means that the physics engine should update an entity.

#### 10.235.3.5 void gazebo::physics::World::EnablePhysicsEngine ( bool *_enable* ) [inline]

enable/disable physics engine during World::Update.

**Parameters**

| | | |
|---|---|---|
| in | *_enable* | True to enable the physics engine. |

#### 10.235.3.6 void gazebo::physics::World::Fini ( )

Finalize the world.

Call this function to tear-down the world.

#### 10.235.3.7 BasePtr gazebo::physics::World::GetByName ( const std::string & *_name* )

Get an element by name.

Searches the list of entities, and return a pointer to the model with a matching _name.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the **Model** (p. 624) to find. |

**Returns**

A pointer to the entity, or NULL if no entity was found.

#### 10.235.3.8 bool gazebo::physics::World::GetEnablePhysicsEngine ( ) [inline]

check if physics engine is enabled/disabled.

**Parameters**

| | |
|---|---|
| *True* | if the physics engine is enabled. |

#### 10.235.3.9 EntityPtr gazebo::physics::World::GetEntity ( const std::string & *_name* )

Get a pointer to an **Entity** (p. 379) based on a name.

This function is the same as GetByName, but limits the search to only Entities.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the **Entity** (p. 379) to find. |

**Returns**

A pointer to the **Entity** (p. 379), or NULL if no **Entity** (p. 379) was found.

#### 10.235.3.10 EntityPtr gazebo::physics::World::GetEntityBelowPoint ( const math::Vector3 & _pt )

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The 3D point to search below |

**Returns**

A pointer to nearest **Entity** (p. 379), NULL if none is found.

#### 10.235.3.11 ModelPtr gazebo::physics::World::GetModel ( unsigned int _index ) const

Get a model based on an index.

Get a **Model** (p. 624) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 1163)

**Parameters**

| | | |
|---|---|---|
| in | *_index* | The index of the model [0..GetModelCount) |

**Returns**

A pointer to the **Model** (p. 624). NULL if _index is invalid.

#### 10.235.3.12 ModelPtr gazebo::physics::World::GetModel ( const std::string & _name )

Get a model by name.

This function is the same as GetByName, but limits the search to only models.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The name of the **Model** (p. 624) to find. |

**Returns**

A pointer to the **Model** (p. 624), or NULL if no model was found.

#### 10.235.3.13 ModelPtr gazebo::physics::World::GetModelBelowPoint ( const math::Vector3 & _pt )

Get the nearest model below a point.

This function makes use of **World::GetEntityBelowPoint** (p. 1162).

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | The 3D point to search below. |

**Returns**

A pointer to nearest **Model** (p. 624), NULL if none is found.

#### 10.235.3.14 unsigned int gazebo::physics::World::GetModelCount ( ) const

Get the number of models.

**Returns**

The number of models in the **World** (p. 1157).

#### 10.235.3.15 Model_V gazebo::physics::World::GetModels ( ) const

Get a list of all the models.

**Returns**

A list of all the Models in the world.

#### 10.235.3.16 std::string gazebo::physics::World::GetName ( ) const

Get the name of the world.

**Returns**

The name of the world.

#### 10.235.3.17 common::Time gazebo::physics::World::GetPauseTime ( ) const

Get the amount of time simulation has been paused.

**Returns**

The pause time.

#### 10.235.3.18 PhysicsEnginePtr gazebo::physics::World::GetPhysicsEngine ( ) const

Return the physics engine.

Get a pointer to the physics engine used by the world.

**Returns**

Pointer to the physics engine.

#### 10.235.3.19 common::Time gazebo::physics::World::GetRealTime ( ) const

Get the real time (elapsed time).

**Returns**

The real time.

#### 10.235.3.20 bool gazebo::physics::World::GetRunning ( ) const

Return the running state of the world.

**Returns**

True if the world is running.

#### 10.235.3.21 EntityPtr gazebo::physics::World::GetSelectedEntity ( ) const

Get the selected **Entity** (p. 379).

The selected entity is set via the GUI.

**Returns**

A point to the **Entity** (p. 379), NULL if nothing is selected.

#### 10.235.3.22 boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex ( ) const [inline]

Get the set world pose mutex.

**Returns**

Pointer to the mutex.

#### 10.235.3.23 common::Time gazebo::physics::World::GetSimTime ( ) const

Get the world simulation time, note if you want the PC wall clock call **common::Time::GetWallTime** (p. 1037).

**Returns**

The current simulation time

#### 10.235.3.24 common::SphericalCoordinatesPtr gazebo::physics::World::GetSphericalCoordinates ( ) const

Return the spherical coordinates converter.

**Returns**

Pointer to the spherical coordinates converter.

#### 10.235.3.25 common::Time gazebo::physics::World::GetStartTime ( ) const

Get the wall time simulation was started.

**Returns**

The start time.

#### 10.235.3.26 void gazebo::physics::World::Init ( )

Initialize the world.

This is called after Load.

#### 10.235.3.27 void gazebo::physics::World::InsertModelFile ( const std::string & *_sdfFilename* )

Insert a model from an SDF file.

Spawns a model into the world base on and SDF file.

**Parameters**

| | | |
|---|---|---|
| in | *_sdfFilename* | The name of the SDF file (including path). |

#### 10.235.3.28 void gazebo::physics::World::InsertModelSDF ( const sdf::SDF & *_sdf* )

Insert a model using SDF.

Spawns a model into the world base on and SDF object.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | A reference to an SDF object. |

#### 10.235.3.29 void gazebo::physics::World::InsertModelString ( const std::string & *_sdfString* )

Insert a model from an SDF string.

Spawns a model into the world base on and SDF string.

**Parameters**

| | | |
|---|---|---|
| in | *_sdfString* | A string containing valid SDF markup. |

#### 10.235.3.30 bool gazebo::physics::World::IsLoaded ( ) const

Return true if the world has been loaded.

**Returns**

True if **World::Load** (p. 1166) has completed.

#### 10.235.3.31 bool gazebo::physics::World::IsPaused ( ) const

Returns the state of the simulation true if paused.

**Returns**

True if paused.

#### 10.235.3.32 void gazebo::physics::World::Load ( sdf::ElementPtr *_sdf* )

Load the world using SDF parameters.

Load a world from and SDF pointer.

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF parameters. |

#### 10.235.3.33 void gazebo::physics::World::LoadPlugin ( const std::string & *_filename,* const std::string & *_name,* sdf::ElementPtr *_sdf* )

Load a plugin.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | The filename of the plugin. |
| in | *_name* | A unique name for the plugin. |
| in | *_sdf* | The SDF to pass into the plugin. |

#### 10.235.3.34 void gazebo::physics::World::PrintEntityTree ( )

Print **Entity** (p. 379) tree.

Prints alls the entities to stdout.

#### 10.235.3.35 void gazebo::physics::World::PublishModelPose ( physics::ModelPtr *_model* )

Publish pose updates for a model.

This list of models to publish is processed and cleared once every iteration.

**Parameters**

| | | |
|---|---|---|
| in | *_model* | Pointer to the model to publish. |

#### 10.235.3.36 void gazebo::physics::World::RemovePlugin ( const std::string & *_name* )

Remove a running plugin.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | The unique name of the plugin to remove. |

#### 10.235.3.37 void gazebo::physics::World::Reset ( )

Reset time and model poses, configurations in simulation.

#### 10.235.3.38 void gazebo::physics::World::ResetEntities ( Base::EntityType *_type* = Base::BASE )

Reset with options.

The _type parameter specifies which type of eneities to reset. See **Base::EntityType** (p. 163).

**Parameters**

| | | |
|---|---|---|
| in | *_type* | The type of reset. |

#### 10.235.3.39 void gazebo::physics::World::ResetTime ( )

Reset simulation time back to zero.

#### 10.235.3.40 void gazebo::physics::World::Run ( unsigned int *_iterations* = 0 )

Run the world in a thread.

Run the update loop.

**Parameters**

| | | |
|---|---|---|
| in | *_iterations* | Run for this many iterations, then stop. A value of zero disables run stop. |

#### 10.235.3.41 void gazebo::physics::World::Save ( const std::string & *_filename* )

Save a world to a file.

Save the current world and its state to a file.

**Parameters**

| | | |
|---|---|---|
| in | *_filename* | Name of the file to save into. |

#### 10.235.3.42 void gazebo::physics::World::SetPaused ( bool *_p* )

Set whether the simulation is paused.

**Parameters**

| | | |
|---|---|---|
| in | *_p* | True pauses the simulation. False runs the simulation. |

#### 10.235.3.43 void gazebo::physics::World::SetSimTime ( const common::Time & _t )

Set the sim time.

**Parameters**

| | | |
|---|---|---|
| in | *_t* | The new simulation time |

#### 10.235.3.44 void gazebo::physics::World::SetState ( const WorldState & _state )

Set the current world state.

**Parameters**

| | |
|---|---|
| *_state* | The state to set the **World** (p. 1157) to. |

#### 10.235.3.45 void gazebo::physics::World::StepWorld ( int _steps )

Step callback.

**Parameters**

| | | |
|---|---|---|
| in | *_steps* | The number of steps the **World** (p. 1157) should take. |

#### 10.235.3.46 void gazebo::physics::World::Stop ( )

Stop the world.

Stop the update loop.

#### 10.235.3.47 std::string gazebo::physics::World::StripWorldName ( const std::string & _name ) const

Return a version of the name with "<world_name>::" removed.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Usually the name of an entity. |

**Returns**

The stripped world name.

#### 10.235.3.48 void gazebo::physics::World::UpdateStateSDF ( )

Update the state SDF value from the current state.

### 10.235.4 Member Data Documentation

#### 10.235.4.1 std::list<Entity*> gazebo::physics::World::dirtyPoses

when physics engine makes an update and changes a link pose, this flag is set to trigger **Entity::SetWorldPose** (p. 388) on the **physics::Link** (p. 542) in World::Update.

The documentation for this class was generated from the following file:

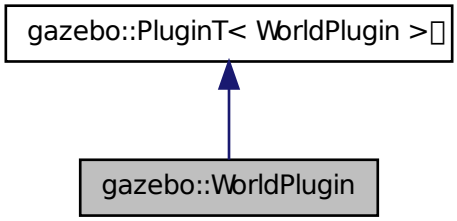
- **World.hh**

## 10.236 gazebo::WorldPlugin Class Reference

A plugin with access to **physics::World** (p. 1157).

`#include <common/common.hh>`

Inheritance diagram for gazebo::WorldPlugin:




### Public Member Functions

- **WorldPlugin** ()

  *Constructor.*
- virtual ~**WorldPlugin** ()

  *Destructor.*
- virtual void **Init** ()
- virtual void **Load** (**physics::WorldPtr** _world, sdf::ElementPtr _sdf)=0

  *Load function.*
- virtual void **Reset** ()

### Additional Inherited Members

### 10.236.1 Detailed Description

A plugin with access to **physics::World** (p. 1157).

See `reference`.

### 10.236.2 Constructor & Destructor Documentation

#### 10.236.2.1 gazebo::WorldPlugin::WorldPlugin ( ) [inline]

Constructor.

References gazebo::PluginT< WorldPlugin >::type, and gazebo::WORLD_PLUGIN.

#### 10.236.2.2 virtual gazebo::WorldPlugin::~WorldPlugin ( ) [inline], [virtual]

Destructor.

### 10.236.3 Member Function Documentation

#### 10.236.3.1 virtual void gazebo::WorldPlugin::Init ( ) [inline], [virtual]

#### 10.236.3.2 virtual void gazebo::WorldPlugin::Load ( physics::WorldPtr *_world,* sdf::ElementPtr *_sdf* ) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer the World |
| in | *_sdf* | Pointer the the SDF element of the plugin. |

#### 10.236.3.3 virtual void gazebo::WorldPlugin::Reset ( ) [inline], [virtual]

The documentation for this class was generated from the following file:

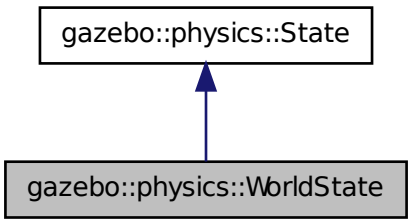
- **Plugin.hh**

## 10.237 gazebo::physics::WorldState Class Reference

Store state information of a **physics::World** (p. 1157) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::WorldState:




## Public Member Functions

- **WorldState** ()

  *Default constructor.*
- **WorldState** (const **WorldPtr** _world)

  *Constructor.*
- **WorldState** (const sdf::ElementPtr _sdf)

  *Constructor.*
- virtual ~**WorldState** ()

  *Destructor.*
- void **FillSDF** (sdf::ElementPtr _sdf)

  *Populate a state SDF element with data from the object.*
- **ModelState GetModelState** (const std::string &_modelName) const

  *Get a model state by model name.*
- unsigned int **GetModelStateCount** () const

  *Get the number of model states.*
- **ModelState_M GetModelStates** (const boost::regex &_regex) const

  *Get model states based on a regular expression.*
- const **ModelState_M** & **GetModelStates** () const

  *Get the model states.*
- bool **HasModelState** (const std::string &_modelName) const

  *Return true if **WorldState** (p. 1170) has a **ModelState** (p. 641) with the given name.*
- bool **IsZero** () const

  *Return true if the values in the state are zero.*
- void **Load** (const **WorldPtr** _world)

  *Load from a **World** (p. 1157) pointer.*
- virtual void **Load** (const sdf::ElementPtr _elem)

  *Load state from SDF element.*
- **WorldState operator+** (const **WorldState** &_state) const

  *Addition operator.*
- **WorldState operator-** (const **WorldState** &_state) const

*Subtraction operator.*

- **WorldState** & **operator=** (const **WorldState** &_state)

  *Assignment operator.*
- virtual void **SetRealTime** (const **common::Time** &_time)

  *Set the real time when this state was generated.*
- virtual void **SetSimTime** (const **common::Time** &_time)

  *Set the sim time when this state was generated.*
- virtual void **SetWallTime** (const **common::Time** &_time)

  *Set the wall time when this state was generated.*
- void **SetWorld** (const **WorldPtr** _world)

  *Set the world.*

## Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::WorldState** &_state)

  *Stream insertion operator.*

## Additional Inherited Members

### 10.237.1 Detailed Description

Store state information of a **physics::World** (p. 1157) object.

Instances of this class contain the state of a **World** (p. 1157) at a specific time. **World** (p. 1157) state includes the state of all models, and their children.

### 10.237.2 Constructor & Destructor Documentation

#### 10.237.2.1 gazebo::physics::WorldState::WorldState ( )

Default constructor.

#### 10.237.2.2 gazebo::physics::WorldState::WorldState ( const WorldPtr *_world* ) [explicit]

Constructor.

Generate a **WorldState** (p. 1170) from an instance of a **World** (p. 1157).

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to a world |

#### 10.237.2.3 gazebo::physics::WorldState::WorldState ( const sdf::ElementPtr *_sdf* ) [explicit]

Constructor.

Build a **WorldState** (p. 1170) from SDF data

**Parameters**

| | | |
|---|---|---|
| in | *_sdf* | SDF data to load a world state from. |

#### 10.237.2.4 virtual gazebo::physics::WorldState::~WorldState ( ) [virtual]

Destructor.

### 10.237.3 Member Function Documentation

#### 10.237.3.1 void gazebo::physics::WorldState::FillSDF ( sdf::ElementPtr *_sdf* )

Populate a state SDF element with data from the object.

**Parameters**

| | | |
|---|---|---|
| out | *_sdf* | SDF element to populate. |

#### 10.237.3.2 ModelState gazebo::physics::WorldState::GetModelState ( const std::string & *_modelName* ) const

Get a model state by model name.

**Parameters**

| | | |
|---|---|---|
| in | *_modelName* | Name of the model state to get. |

**Returns**

The model state.

**Exceptions**

| | |
|---|---|
| ***common::Exception*** *(p. 416)* | When the _modelName doesn't exist. |

#### 10.237.3.3 unsigned int gazebo::physics::WorldState::GetModelStateCount ( ) const

Get the number of model states.

Returns the number of models in this instance.

**Returns**

Number of models.

#### 10.237.3.4 ModelState_M gazebo::physics::WorldState::GetModelStates ( const boost::regex & *_regex* ) const

Get model states based on a regular expression.

**Parameters**

| | | |
|---|---|---|
| in | *_regex* | The regular expression. |

**Returns**

List of model states whose names match the regular expression.

#### 10.237.3.5 const ModelState_M& gazebo::physics::WorldState::GetModelStates ( ) const

Get the model states.

**Returns**

A vector of model states.

#### 10.237.3.6 bool gazebo::physics::WorldState::HasModelState ( const std::string & *_modelName* ) const

Return true if **WorldState** (p. 1170) has a **ModelState** (p. 641) with the given name.

**Parameters**

| | | |
|---|---|---|
| in | *_modelName* | Name of the model to search for. |

**Returns**

True if the **ModelState** (p. 641) exists.

#### 10.237.3.7 bool gazebo::physics::WorldState::IsZero ( ) const

Return true if the values in the state are zero.

This will check to see if the all model states are zero.

**Returns**

True if the values in the state are zero.

#### 10.237.3.8 void gazebo::physics::WorldState::Load ( const WorldPtr *_world* )

Load from a **World** (p. 1157) pointer.

Generate a **WorldState** (p. 1170) from an instance of a **World** (p. 1157).

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to a world |

#### 10.237.3.9 virtual void gazebo::physics::WorldState::Load ( const sdf::ElementPtr _elem ) [virtual]

Load state from SDF element.

Set a **WorldState** (p. 1170) from an SDF element containing **WorldState** (p. 1170) info.

**Parameters**

| | | |
|---|---|---|
| in | *_elem* | Pointer to the **WorldState** (p. 1170) SDF element. |

Reimplemented from **gazebo::physics::State** (p. 1000).

#### 10.237.3.10 WorldState gazebo::physics::WorldState::operator+ ( const WorldState & _state ) const

Addition operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to add. |

**Returns**

The resulting state.

#### 10.237.3.11 WorldState gazebo::physics::WorldState::operator- ( const WorldState & _state ) const

Subtraction operator.

**Parameters**

| | | |
|---|---|---|
| in | *_pt* | A state to substract. |

**Returns**

The resulting state.

#### 10.237.3.12 WorldState& gazebo::physics::WorldState::operator= ( const WorldState & _state )

Assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *_state* | **State** (p. 998) value |

**Returns**

Reference to this

#### 10.237.3.13 virtual void gazebo::physics::WorldState::SetRealTime ( const common::Time & _time ) [virtual]

Set the real time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Clock time since simulation was stated. |

Reimplemented from **gazebo::physics::State** (p. 1001).

#### 10.237.3.14 virtual void gazebo::physics::WorldState::SetSimTime ( const common::Time & *_time* ) [virtual]

Set the sim time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | Simulation time when the data was recorded. |

Reimplemented from **gazebo::physics::State** (p. 1001).

#### 10.237.3.15 virtual void gazebo::physics::WorldState::SetWallTime ( const common::Time & *_time* ) [virtual]

Set the wall time when this state was generated.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | The absolute clock time when the **State** (p. 998) data was recorded. |

Reimplemented from **gazebo::physics::State** (p. 1002).

#### 10.237.3.16 void gazebo::physics::WorldState::SetWorld ( const WorldPtr *_world* )

Set the world.

**Parameters**

| | | |
|---|---|---|
| in | *_world* | Pointer to the world. |

### 10.237.4 Friends And Related Function Documentation

#### 10.237.4.1 std::ostream& operator<< ( std::ostream & *_out,* const gazebo::physics::WorldState & *_state* ) [friend]

Stream insertion operator.

**Parameters**

| | | |
|---|---|---|
| in | *_out* | output stream |
| in | *_state* | **World** (p. 1157) state to output |

**Returns**

the stream

The documentation for this class was generated from the following file:

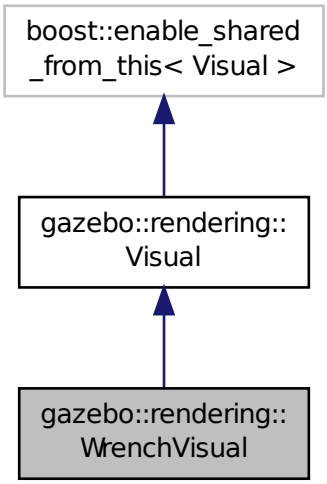
- **WorldState.hh**

## 10.238 gazebo::rendering::WrenchVisual Class Reference

Visualization for sonar data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::WrenchVisual:




### Public Member Functions

- **WrenchVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  *Constructor.*
- virtual ~**WrenchVisual** ()

  *Destructor.*
- void **Load** (ConstJointPtr &_msg)

  *Load the visual based on a message.*
- void **SetEnabled** (bool _enabled)

  *Set to true to enable wrench visualization.*

### Additional Inherited Members

### 10.238.1 Detailed Description

Visualization for sonar data.

### 10.238.2 Constructor & Destructor Documentation

#### 10.238.2.1 gazebo::rendering::WrenchVisual::WrenchVisual ( const std::string & _name, VisualPtr _vis, const std::string & _topicName )

Constructor.

**Parameters**

| | | |
|---|---|---|
| in | *_name* | Name of the visual. |
| in | *_vis* | Pointer to the parent **Visual** (p. 1121). |
| in | *_topicName* | Name of the topic that has sonar data. |

#### 10.238.2.2 virtual gazebo::rendering::WrenchVisual::~WrenchVisual ( ) [virtual]

Destructor.

### 10.238.3 Member Function Documentation

#### 10.238.3.1 void gazebo::rendering::WrenchVisual::Load ( ConstJointPtr & _msg )

Load the visual based on a message.

**Parameters**

| | | |
|---|---|---|
| in | *_msg* | Joint message |

#### 10.238.3.2 void gazebo::rendering::WrenchVisual::SetEnabled ( bool _enabled )

Set to true to enable wrench visualization.

**Parameters**

| | | |
|---|---|---|
| in | *_enabled* | True to show wrenches, false to hide. |

The documentation for this class was generated from the following file:

- **WrenchVisual.hh**

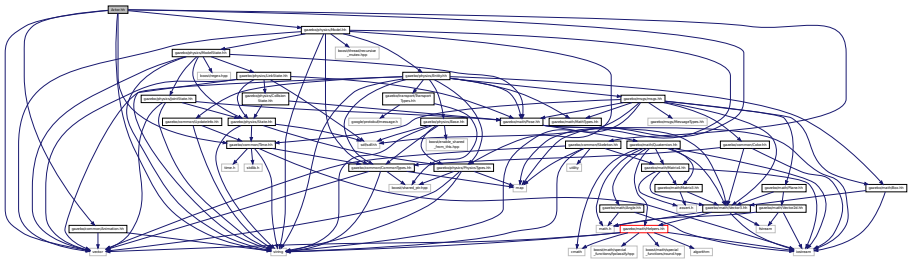
# Chapter 11

# File Documentation

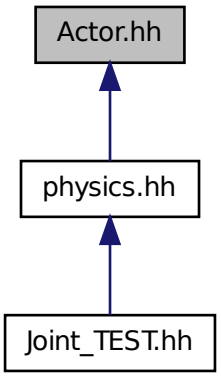
## 11.1 Actor.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/physics/Model.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/common/Skeleton.hh"
#include "gazebo/common/Animation.hh"
```

Include dependency graph for Actor.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::Actor**

  ***Actor*** *(*p. *131) class enables GPU based mesh model / skeleton scriptable animation.*

- struct **gazebo::physics::TrajectoryInfo**

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::common**

  *Common namespace.*

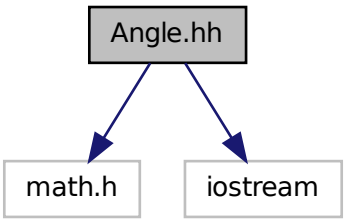
- namespace **gazebo::physics**

  *namespace for physics*

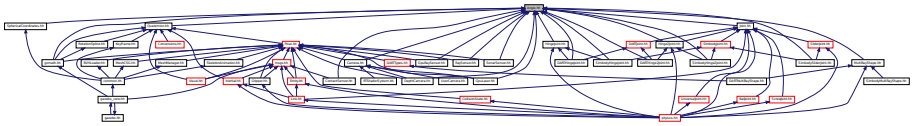
## 11.2 Angle.hh File Reference

```
#include <math.h>
#include <iostream>
```

Include dependency graph for Angle.hh:




This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::math::Angle**

  *An angle and related functions.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

## Macros

- #define **GZ_DTOR**(d) ((d) * M_PI / 180)

  *Converts degrees to radians.*
- #define **GZ_NORMALIZE**(a) (atan2(sin(a), cos(a)))

  *Macro tha normalizes an angle in the range -Pi to Pi.*
- #define **GZ_RTOD**(r) ((r) * 180 / M_PI)

  *Macro that converts radians to degrees.*

### 11.2.1 Macro Definition Documentation

#### 11.2.1.1 #define GZ_DTOR( *d* ) ((d) * M_PI / 180)

Converts degrees to radians.

**Parameters**

| | | |
|---|---|---|
| in | *degrees* | |

**Returns**

radians

#### 11.2.1.2 #define GZ_NORMALIZE( *a* ) (atan2(sin(a), cos(a)))

Macro tha normalizes an angle in the range -Pi to Pi.

**Parameters**

| | | |
|---|---|---|
| in | *angle* | |

**Returns**

the angle, in range

#### 11.2.1.3 #define GZ_RTOD( *r* ) ((r) * 180 / M_PI)

Macro that converts radians to degrees.

**Parameters**

| | | |
|---|---|---|
| in | *radians* | |

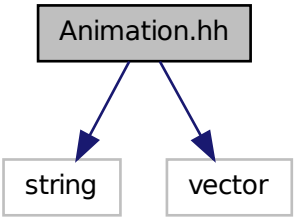
**Returns**

degrees

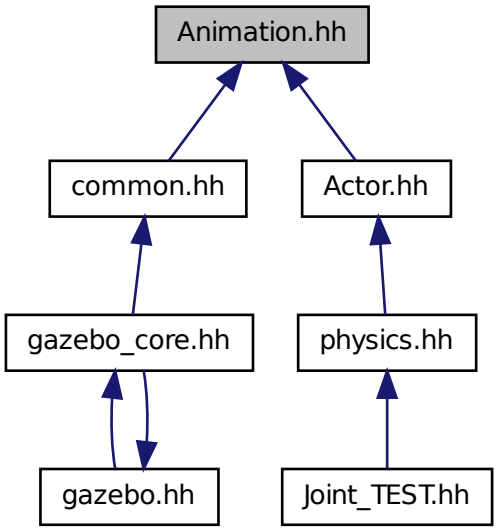
## 11.3 Animation.hh File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for Animation.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::Animation**

  *Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.*
- class **gazebo::common::NumericAnimation**

  *A numeric animation.*
- class **gazebo::common::PoseAnimation**

  *A pose animation.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

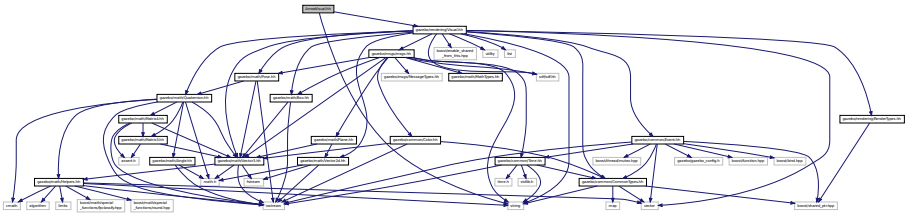
  *Common namespace.*
- namespace **gazebo::math**

  *Math namespace.*

## 11.4 ArrowVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for ArrowVisual.hh:



### Classes

- class **gazebo::rendering::ArrowVisual**

  *Basic arrow visualization.*

### Namespaces

- namespace **gazebo**

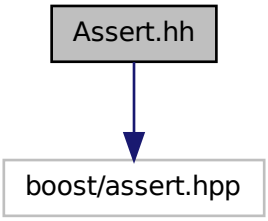
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*
- namespace **ogre**

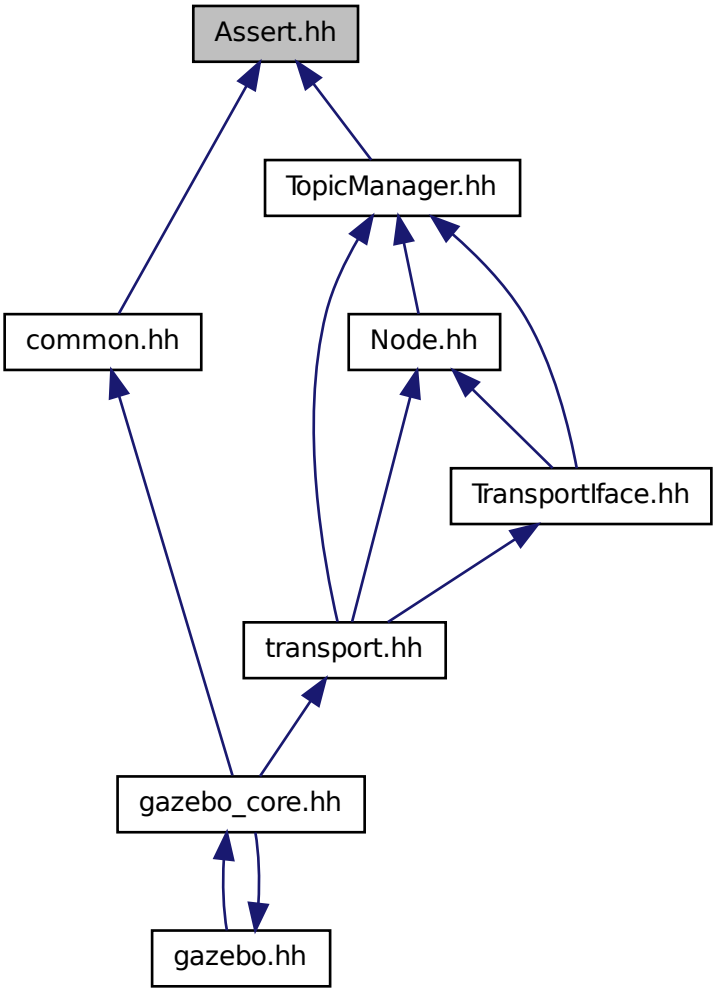
## 11.5 Assert.hh File Reference

```
#include <boost/assert.hpp>
```

Include dependency graph for Assert.hh:



This graph shows which files directly or indirectly include this file:




## Macros

- #define **GZ_ASSERT**(_expr, _msg) BOOST_ASSERT_MSG(_expr, _msg)

  *This macro define the standard way of launching an exception inside gazebo.*

### 11.5.1 Macro Definition Documentation

#### 11.5.1.1 #define GZ_ASSERT( *_expr, _msg* ) BOOST_ASSERT_MSG(_expr, _msg)

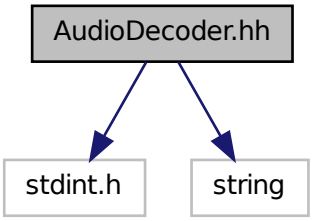
This macro define the standard way of launching an exception inside gazebo.

Referenced by gazebo::transport::TopicManager::Advertise().

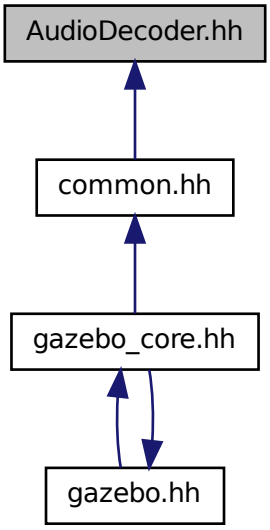
## 11.6 AudioDecoder.hh File Reference

```
#include <stdint.h>
#include <string>
```

Include dependency graph for AudioDecoder.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::AudioDecoder**

  *An audio decoder based on FFMPEG.*

### Namespaces

- namespace **gazebo**

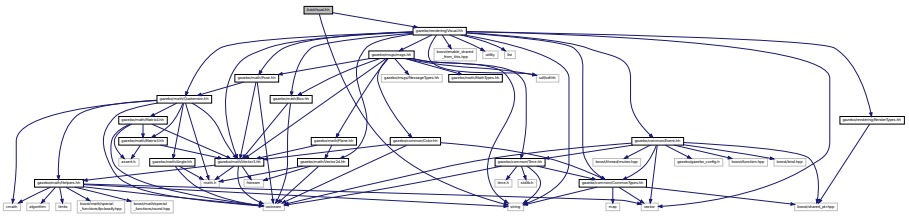
  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## 11.7 AxisVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for AxisVisual.hh:



### Classes

- class **gazebo::rendering::AxisVisual**

  *Basic axis visualization.*

### Namespaces

- namespace **gazebo**

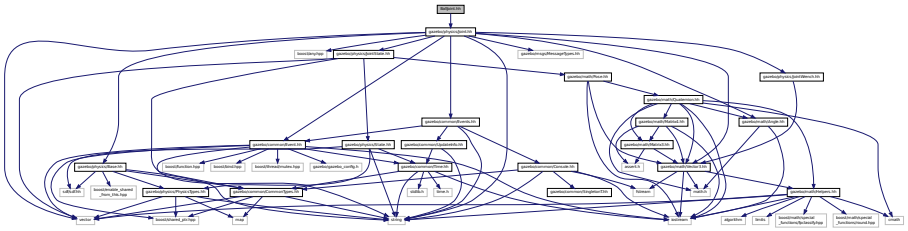
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

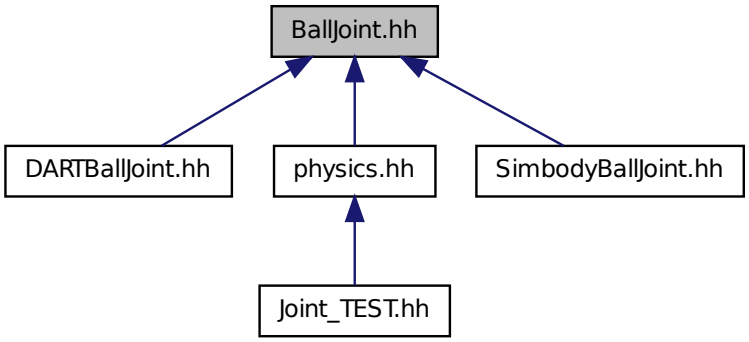
## 11.8 BallJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for BallJoint.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::physics::BallJoint**< **T** >

  ***Base*** (p. *159*) *class for a ball joint.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

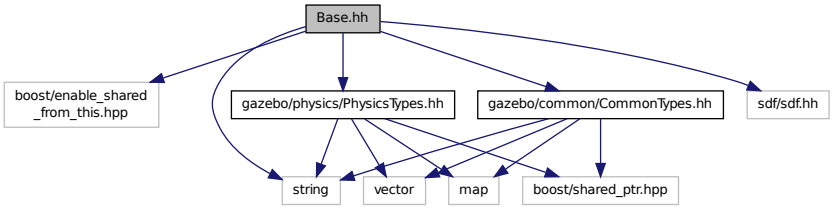
  *namespace for physics*

# 11.9 Base.hh File Reference

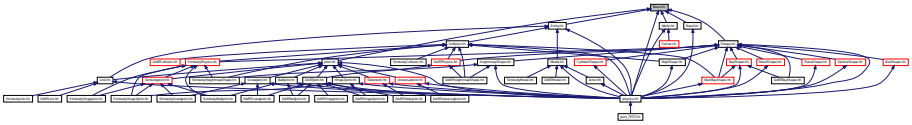
```
#include <boost/enable_shared_from_this.hpp>
```

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for Base.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::physics::Base**

  ***Base*** *(p. 159) class for most physics classes.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

### Variables

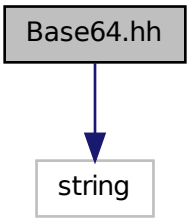
- static std::string **gazebo::physics::EntityTypename** [ ]

  *String names for the different entity types.*

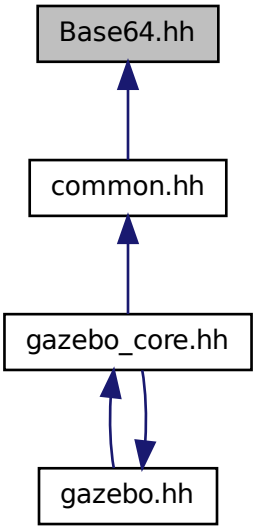
## 11.10 Base64.hh File Reference

```
#include <string>
```

Include dependency graph for Base64.hh:




This graph shows which files directly or indirectly include this file:




## Functions

- std::string **Base64Decode** (const std::string &_encodedString)

  *Decode a base64 string.*

- void **Base64Encode** (const char *_bytesToEncode, unsigned int _len, std::string &_result)

  *Encode a binary string into base 64.*

### 11.10.1 Function Documentation

#### 11.10.1.1 std::string Base64Decode ( const std::string & *_encodedString* )

Decode a base64 string.

**Parameters**

| | | |
|---|---|---|
| in | *_encodedString* | A base 64 encoded string. |

**Returns**

The decoded string.

#### 11.10.1.2 void Base64Encode ( const char * *_bytesToEncode*, unsigned int *_len*, std::string & *_result* )

Encode a binary string into base 64.

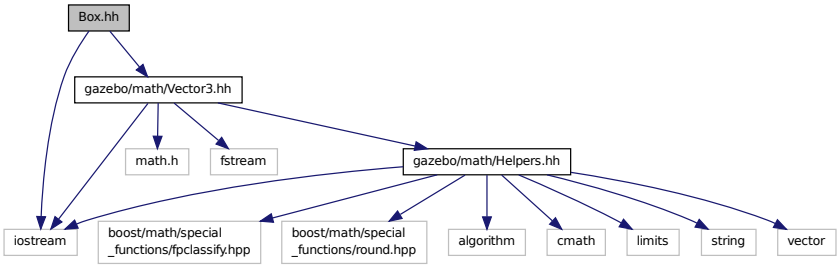
**Parameters**

| | | |
|---|---|---|
| in | *_bytesToEncode* | String of bytes to encode. |
| in | *_len* | Length of _bytesToEncode. |
| out | *_result* | Based64 string is appended to this string. |

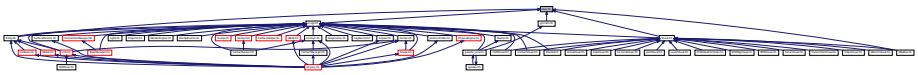
## 11.11 Box.hh File Reference

```
#include <iostream>
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for Box.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::math::Box**

  *Mathematical representation of a box and related functions.*

### Namespaces

- namespace **gazebo**

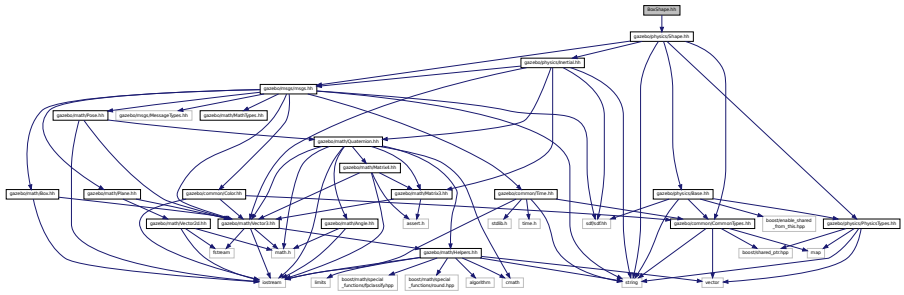
  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

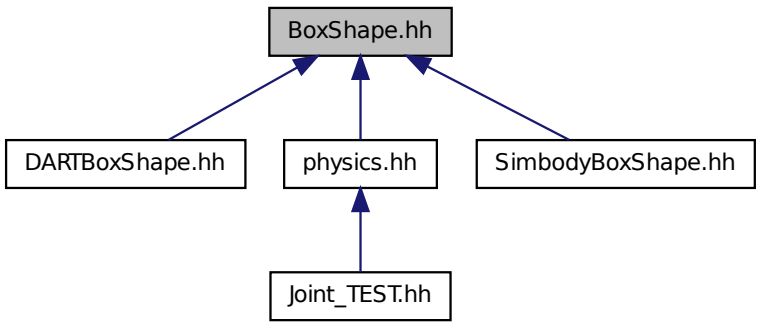
## 11.12 BoxShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for BoxShape.hh:



This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::physics::BoxShape**

  *Box geometry primitive.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

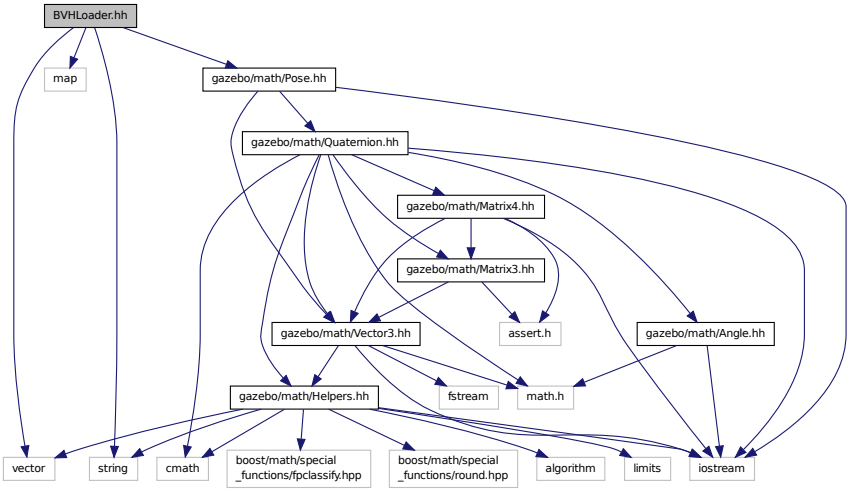
- namespace **gazebo::physics**

  *namespace for physics*

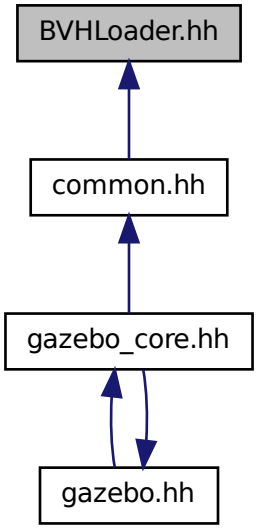
## 11.13 BVHLoader.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include "gazebo/math/Pose.hh"
```

Include dependency graph for BVHLoader.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::common::BVHLoader**

  *Handles loading BVH animation files.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## Macros

- #define **X_POSITION** 0
- #define **X_ROTATION** 3
- #define **Y_POSITION** 1
- #define **Y_ROTATION** 4
- #define **Z_POSITION** 2
- #define **Z_ROTATION** 5

### 11.13.1 Macro Definition Documentation

#### 11.13.1.1 #define X_POSITION 0

#### 11.13.1.2 #define X_ROTATION 3

#### 11.13.1.3 #define Y_POSITION 1

#### 11.13.1.4 #define Y_ROTATION 4

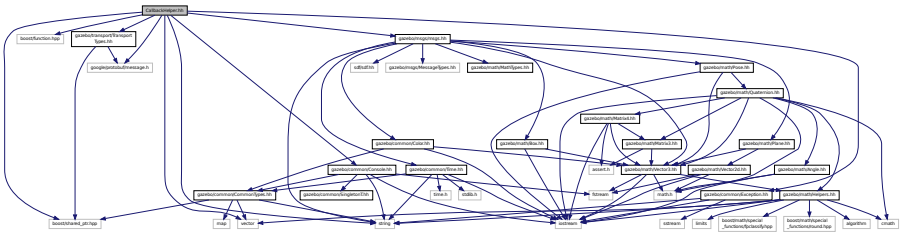
#### 11.13.1.5 #define Z_POSITION 2

#### 11.13.1.6 #define Z_ROTATION 5

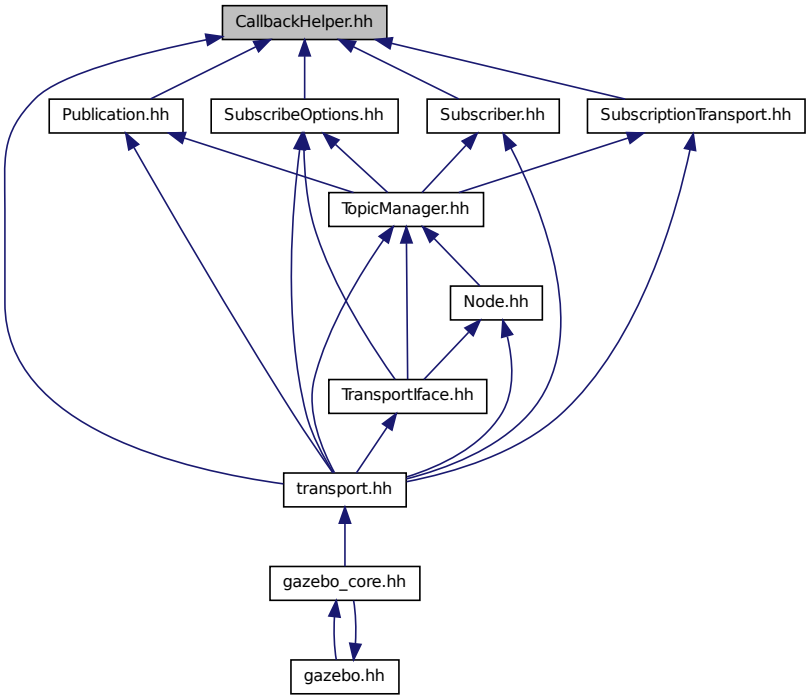
## 11.14 CallbackHelper.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <vector>
#include <string>
#include "gazebo/common/Console.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for CallbackHelper.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::CallbackHelper**

  *A helper class to handle callbacks when messages arrive.*
- class **gazebo::transport::CallbackHelperT< M >**

  *Callback helper Template.*
- class **gazebo::transport::RawCallbackHelper**

  *Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

## Typedefs

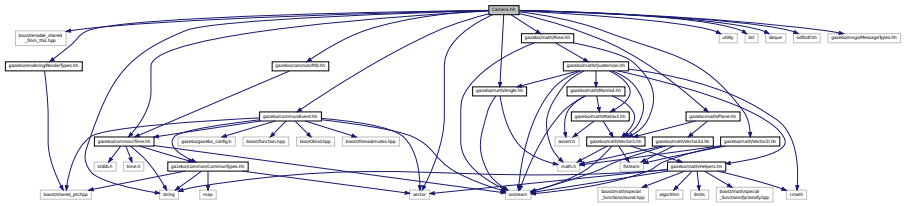
- typedef boost::shared_ptr
  < CallbackHelper > **gazebo::transport::CallbackHelperPtr**

  *boost shared pointer to* ***transport::CallbackHelper*** *(*p. *180)*

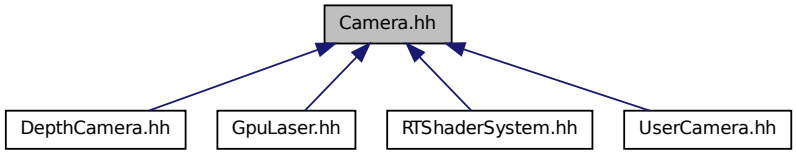
## 11.15 Camera.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <deque>
#include <sdf/sdf.hh>
#include "gazebo/common/Event.hh"
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/math/Vector2i.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for Camera.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::rendering::Camera**

  *Basic camera sensor.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

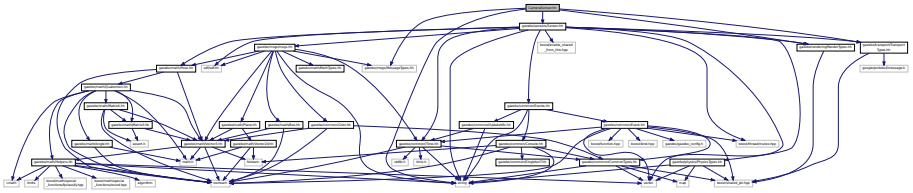
- namespace **gazebo::rendering**

  *Rendering namespace.*
- namespace **Ogre**

## 11.16 CameraSensor.hh File Reference

```
#include <string>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for CameraSensor.hh:



### Classes

- class **gazebo::sensors::CameraSensor**

  *Basic camera sensor.*

### Namespaces

- namespace **gazebo**

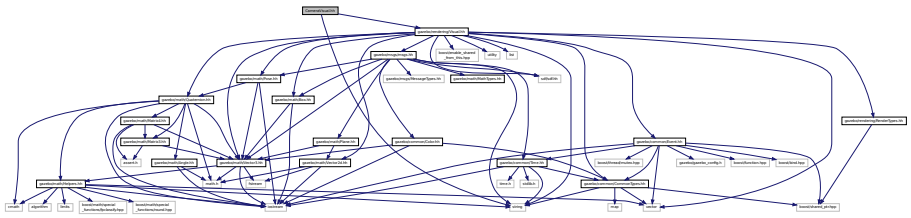
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.17 CameraVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for CameraVisual.hh:



### Classes

- class **gazebo::rendering::CameraVisual**

  *Basic camera visualization.*

### Namespaces

- namespace **gazebo**

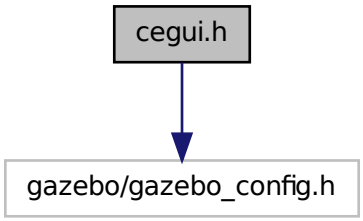
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

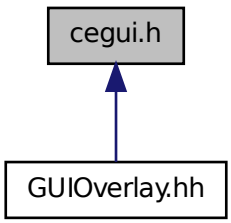
## 11.18 cegui.h File Reference

```
#include "gazebo/gazebo_config.h"
```

Include dependency graph for cegui.h:



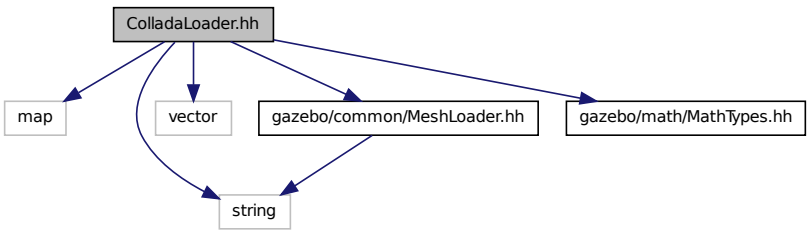

This graph shows which files directly or indirectly include this file:



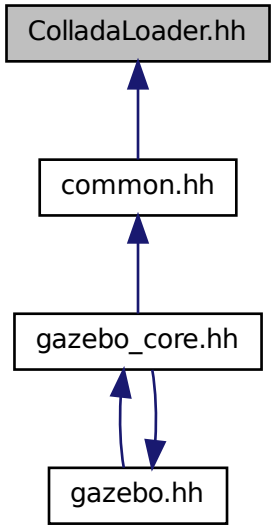
## 11.19 ColladaLoader.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/MeshLoader.hh"
#include "gazebo/math/MathTypes.hh"
```

Include dependency graph for ColladaLoader.hh:




This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::ColladaLoader**

  *Class used to load Collada mesh files.*

### Namespaces

- namespace **gazebo**

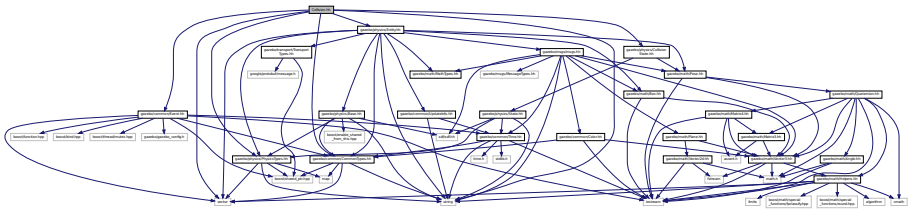
  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## 11.20 Collision.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/Event.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/CollisionState.hh"
#include "gazebo/physics/Entity.hh"
```

Include dependency graph for Collision.hh:



This graph shows which files directly or indirectly include this file:

### Classes

- class **gazebo::physics::Collision**

  ***Base*** *(p. 159) class for all collision entities.*

### Namespaces

- namespace **gazebo**

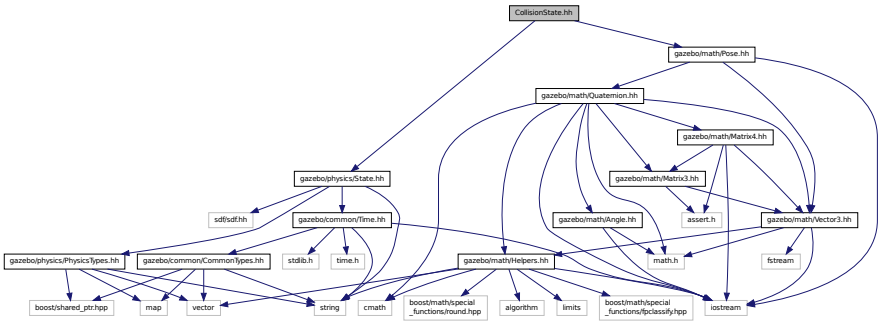
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

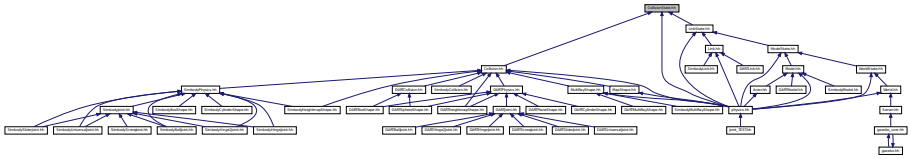
## 11.21 CollisionState.hh File Reference

```
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.hh"
```

Include dependency graph for CollisionState.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::physics::CollisionState**

  *Store state information of a **physics::Collision*** (p. *220*) *object.*

### Namespaces

- namespace **gazebo**

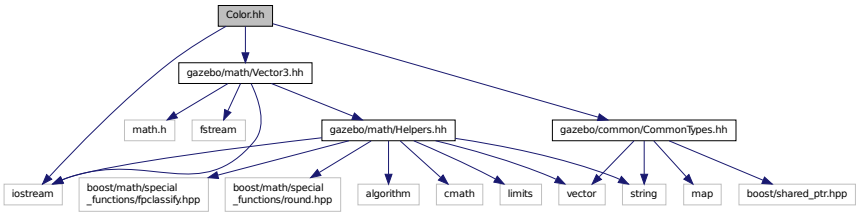
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.22 Color.hh File Reference

```
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for Color.hh:




This graph shows which files directly or indirectly include this file:

## Classes

- class **gazebo::common::Color**

  *Defines a color.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

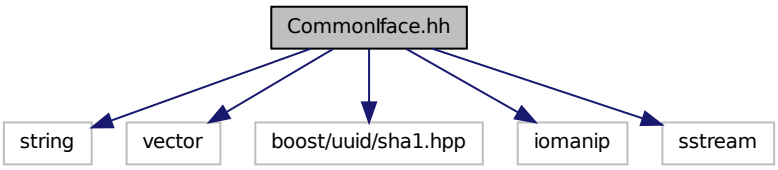
- namespace **gazebo::common**

  *Common namespace.*

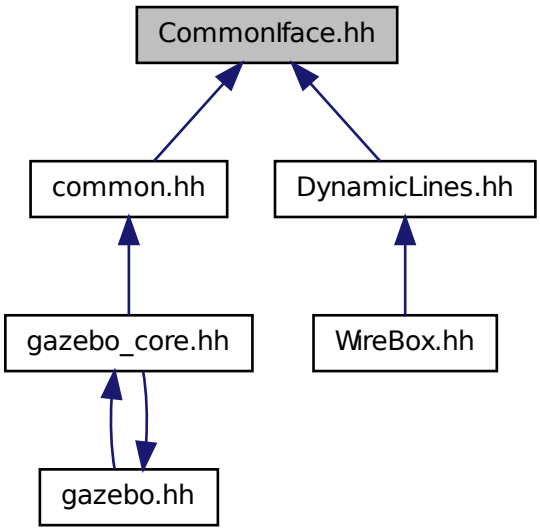
# 11.23 CommonIface.hh File Reference

```
#include <string>
#include <vector>
#include <boost/uuid/sha1.hpp>
#include <iomanip>
#include <sstream>
```

Include dependency graph for CommonIface.hh:




This graph shows which files directly or indirectly include this file:




## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## Functions

- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)

*add path sufix to **common::SystemPaths** (p. 1025)*

- std::string **gazebo::common::find_file** (const std::string &_file)

  *search for file in **common::SystemPaths** (p. 1025)*

- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)

  *search for file in **common::SystemPaths** (p. 1025)*

- std::string **gazebo::common::find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths** (p. 1025)*

- template<typename T >
  std::string **gazebo::common::get_sha1** (const T &_buffer)

  *Compute the SHA1 hash of an array of bytes.*

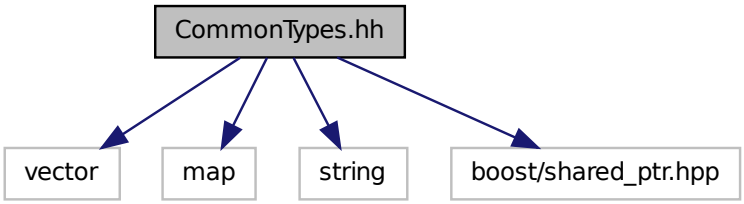
- void **gazebo::common::load** ()

  *Load the common library.*

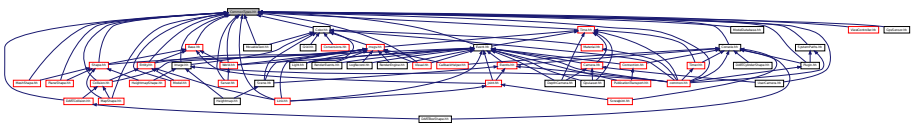
## 11.24 CommonTypes.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for CommonTypes.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::ParamT**< **T** >

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*
- namespace **gazebo::event**

  ***Event*** *(p. 390) namespace.*

## Macros

- #define **GAZEBO_DEPRECATED**(version) ()
- #define **GAZEBO_FORCEINLINE**
- #define **NULL** 0

## Typedefs

- typedef boost::shared_ptr < Animation > **gazebo::common::AnimationPtr**
- typedef std::vector < ConnectionPtr > **gazebo::event::Connection_V**
- typedef boost::shared_ptr < Connection > **gazebo::event::ConnectionPtr**
- typedef boost::shared_ptr < DiagnosticTimer > **gazebo::common::DiagnosticTimerPtr**
- typedef boost::shared_ptr < GUIPlugin > **gazebo::GUIPluginPtr**
- typedef boost::shared_ptr < ModelPlugin > **gazebo::ModelPluginPtr**
- typedef boost::shared_ptr < NumericAnimation > **gazebo::common::NumericAnimationPtr**
- typedef std::vector < common::Param * > **gazebo::common::Param_V**
- typedef boost::shared_ptr < PoseAnimation > **gazebo::common::PoseAnimationPtr**
- typedef boost::shared_ptr < SensorPlugin > **gazebo::SensorPluginPtr**
- typedef boost::shared_ptr < SphericalCoordinates > **gazebo::common::SphericalCoordinatesPtr**
- typedef std::map< std::string, std::string > **gazebo::common::StrStr_M**
- typedef boost::shared_ptr < SystemPlugin > **gazebo::SystemPluginPtr**
- typedef boost::shared_ptr < VisualPlugin > **gazebo::VisualPluginPtr**
- typedef boost::shared_ptr < WorldPlugin > **gazebo::WorldPluginPtr**

### Variables

- static const double **gazebo::common::SpeedOfLight** = 299792458

  *Speed of light.*

### 11.24.1 Macro Definition Documentation

#### 11.24.1.1 #define GAZEBO_DEPRECATED( *version* ) ()

#### 11.24.1.2 #define GAZEBO_FORCEINLINE

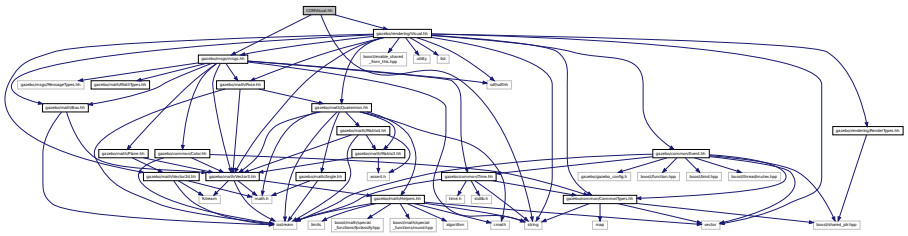
#### 11.24.1.3 #define NULL 0

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::event::EventT< T >::Disconnect(), gazebo::common::get_sha1(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), gazebo::transport::SubscribeOptions::Init(), gazebo::PluginT< ModelPlugin >::PluginT(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::DARTBoxShape::SetSize(), and Joint_TEST::SpawnJoint().

## 11.25 COMVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/msgs.hh"
```

Include dependency graph for COMVisual.hh:



### Classes

- class **gazebo::rendering::COMVisual**

  *Basic Center of Mass visualization.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

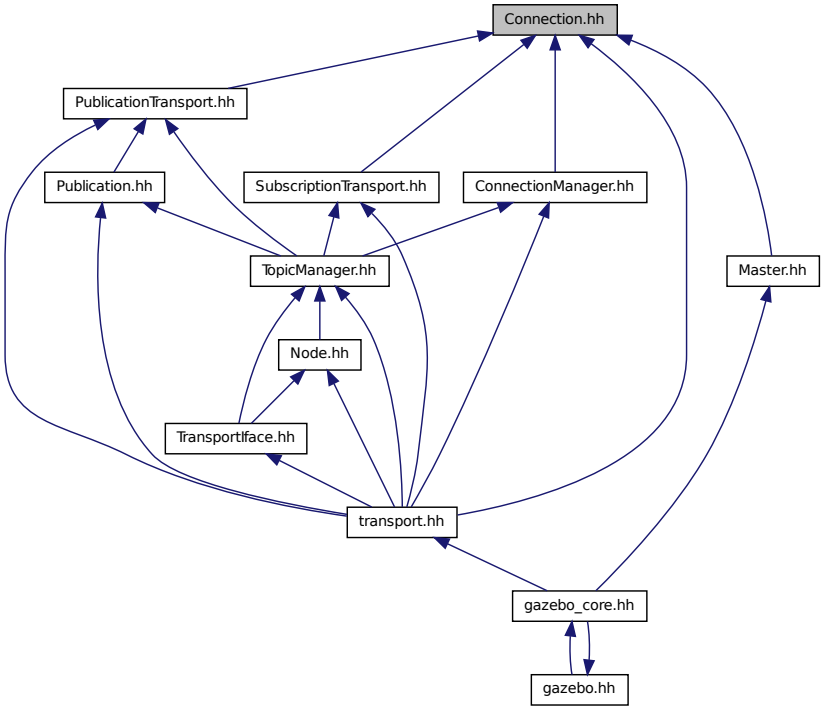
  *Rendering namespace.*
- namespace **ogre**

## 11.26 Connection.hh File Reference

```
#include <tbb/task.h>
#include <google/protobuf/message.h>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include <boost/thread.hpp>
#include <boost/tuple/tuple.hpp>
#include <string>
#include <vector>
#include <iostream>
#include <iomanip>
#include <deque>
#include <utility>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Console.hh"
#include "gazebo/common/Exception.hh"
```

Include dependency graph for Connection.hh:

This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::Connection**

  *Single TCP/IP connection manager.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

## Macros

- #define **HEADER_LENGTH** 8

## Typedefs

- typedef boost::shared_ptr
  < Connection > **gazebo::transport::ConnectionPtr**

### Functions

- bool **gazebo::transport::is_stopped** ()

  *Is the transport system stopped?*

### 11.26.1 Macro Definition Documentation

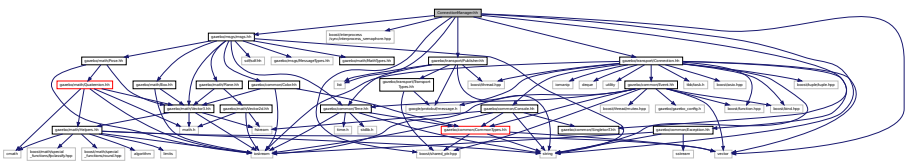
#### 11.26.1.1 #define HEADER_LENGTH 8

Referenced by gazebo::transport::Connection::AsyncRead().

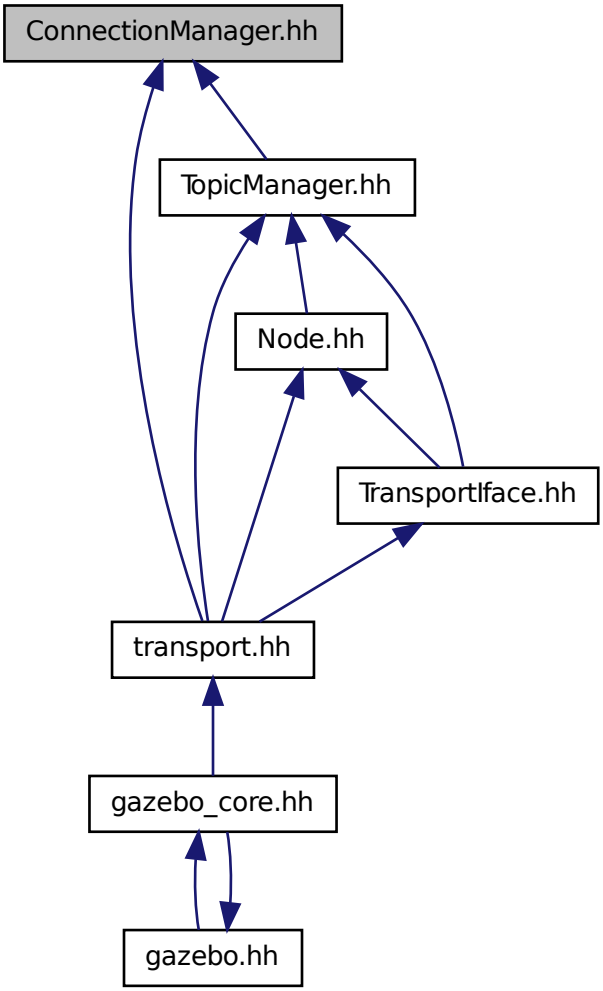
## 11.27 ConnectionManager.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <boost/interprocess/sync/interprocess_semaphore.hpp>
#include <string>
#include <list>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/transport/Publisher.hh"
#include "gazebo/transport/Connection.hh"
```

Include dependency graph for ConnectionManager.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::ConnectionManager**

  *Manager of connections.*

## Namespaces

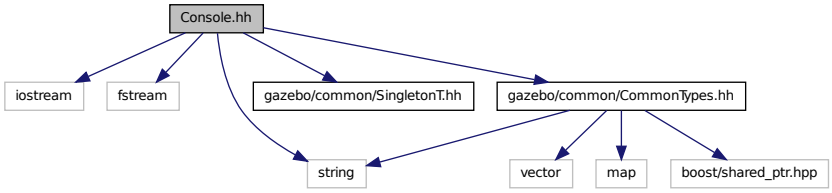
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

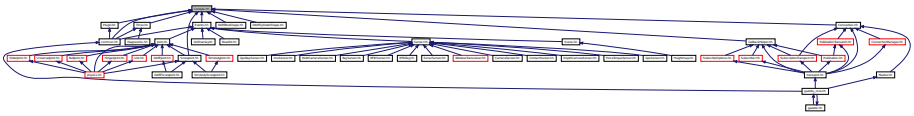
## 11.28 Console.hh File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Console.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::Console**

  *Message, error, warning functionality.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

### Macros

- #define **gzclr_end** "\033[0m"

  *End marker.*
- #define **gzclr_start**(clr) "\033[1;33m"

  *Start marker.*
- #define **gzdbg** (**gazebo::common::Console::Instance**()->ColorMsg("Dbg", 36))

  *Output a debug message.*
- #define **gzerr**

*Output an error message.*

- #define **gzlog** (**gazebo::common::Console::Instance**()->Log())

  *Output a message to a log file.*

- #define **gzmsg** (**gazebo::common::Console::Instance**()->ColorMsg("Msg", 32))

  *Output a message.*

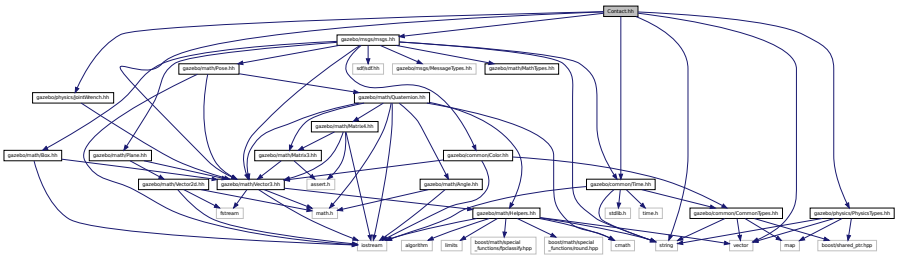
- #define **gzwarn**

  *Output a warning message.*

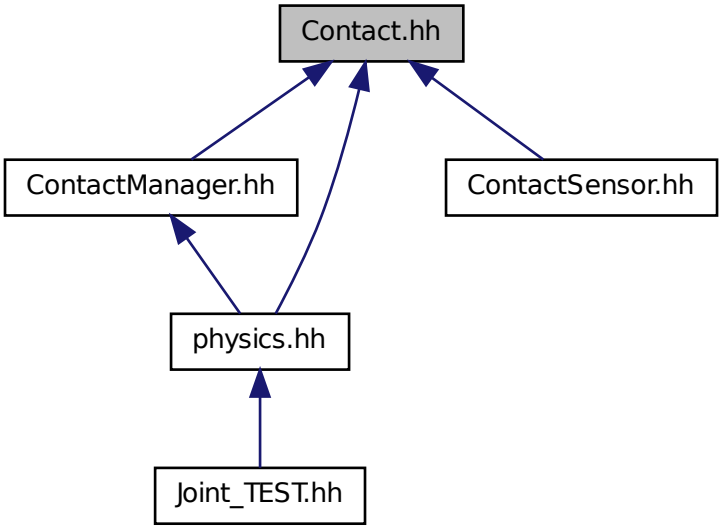
## 11.29 Contact.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/JointWrench.hh"
```

Include dependency graph for Contact.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::physics::Contact**

  *A contact between two collisions.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## Macros

- #define **MAX_COLLIDE_RETURNS** 250
- #define **MAX_CONTACT_JOINTS** 32

### 11.29.1 Macro Definition Documentation

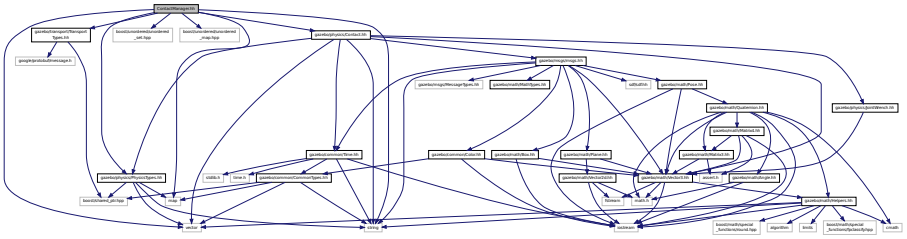
#### 11.29.1.1 #define MAX_COLLIDE_RETURNS 250

#### 11.29.1.2 #define MAX_CONTACT_JOINTS 32

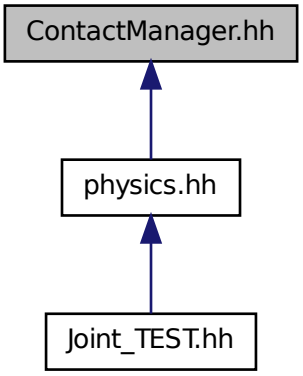
## 11.30 ContactManager.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <boost/unordered/unordered_set.hpp>
#include <boost/unordered/unordered_map.hpp>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Contact.hh"
```

Include dependency graph for ContactManager.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::ContactManager**

  *Aggregates all the contact information generated by the collision detection engine.*

- class **gazebo::physics::ContactPublisher**

  *A custom contact publisher created for each contact filter in the **Contact** (p. 260) Manager.*

### Namespaces

- namespace **gazebo**

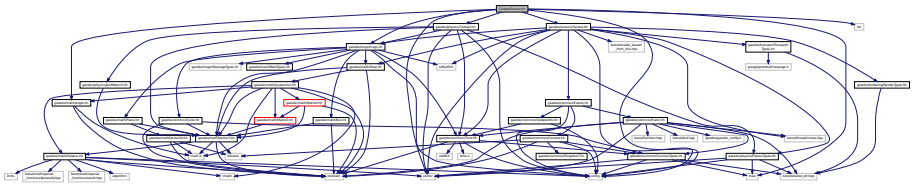
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.31 ContactSensor.hh File Reference

```
#include <vector>
#include <map>
#include <list>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/physics/Contact.hh"
```

Include dependency graph for ContactSensor.hh:



### Classes

- class **gazebo::sensors::ContactSensor**

  *Contact sensor.*

### Namespaces

- namespace **gazebo**

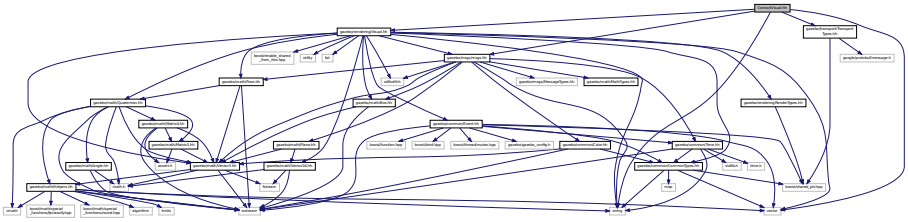
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.32 ContactVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for ContactVisual.hh:



### Classes

- class **gazebo::rendering::ContactVisual**

  *Contact visualization.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::rendering**

  *Rendering namespace.*

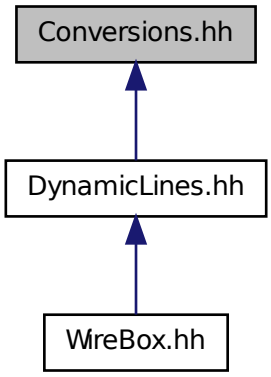
- namespace **Ogre**

## 11.33 Conversions.hh File Reference

```
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Color.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh"
```

Include dependency graph for Conversions.hh:

This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::rendering::Conversions**

  ***Conversions*** *(*p. *273)* ***Conversions.hh*** *(*p. *1218)* ***rendering/Conversions.hh*** *(*p. *1218)*.

### Namespaces

- namespace **gazebo**

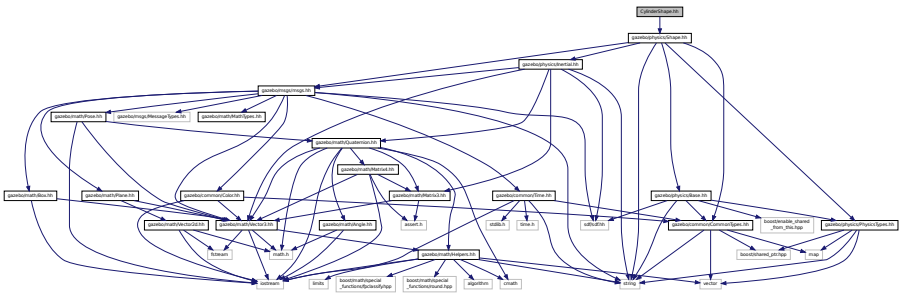
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

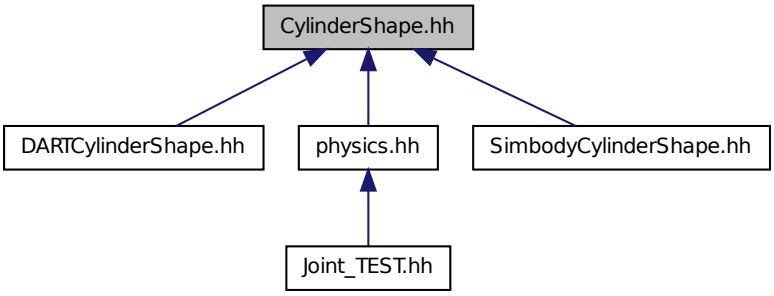
## 11.34 CylinderShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for CylinderShape.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::CylinderShape**

  *Cylinder collision.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::physics**

  *namespace for physics*

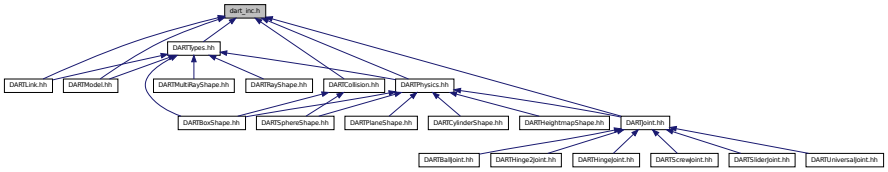
## 11.35 dart_inc.h File Reference

```
#include <dart/math/Helpers.h>
```

```
#include <dart/math/Geometry.h>
#include <dart/collision/CollisionDetector.h>
#include <dart/collision/dart/DARTCollisionDetector.h>
#include <dart/integration/Integrator.h>
#include <dart/integration/EulerIntegrator.h>
#include <dart/integration/RK4Integrator.h>
#include <dart/dynamics/BallJoint.h>
#include <dart/dynamics/BodyNode.h>
#include <dart/dynamics/BoxShape.h>
#include <dart/dynamics/CylinderShape.h>
#include <dart/dynamics/EllipsoidShape.h>
#include <dart/dynamics/FreeJoint.h>
#include <dart/dynamics/GenCoord.h>
#include <dart/dynamics/Joint.h>
#include <dart/dynamics/MeshShape.h>
#include <dart/dynamics/PrismaticJoint.h>
#include <dart/dynamics/RevoluteJoint.h>
#include <dart/dynamics/Shape.h>
#include <dart/dynamics/Skeleton.h>
#include <dart/dynamics/ScrewJoint.h>
#include <dart/dynamics/UniversalJoint.h>
#include <dart/dynamics/WeldJoint.h>
#include <dart/constraint/Constraint.h>
#include <dart/constraint/ConstraintDynamics.h>
#include <dart/simulation/World.h>
```

Include dependency graph for dart_inc.h:

This graph shows which files directly or indirectly include this file:




## 11.36 DARTBallJoint.hh File Reference

```
#include "gazebo/physics/BallJoint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
```

Include dependency graph for DARTBallJoint.hh:

### Classes

- class **gazebo::physics::DARTBallJoint**

  *An **DARTBallJoint** (p. 279).*

### Namespaces

- namespace **gazebo**

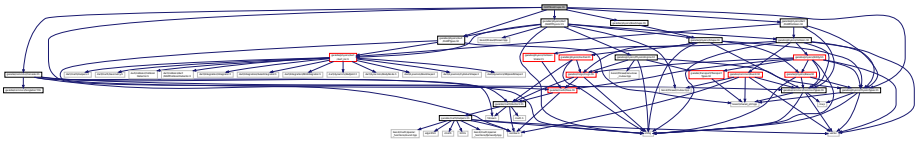
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.37 DARTBoxShape.hh File Reference

```
#include "gazebo/common/Console.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/physics/dart/DARTCollision.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/BoxShape.hh"
```

Include dependency graph for DARTBoxShape.hh:



### Classes

- class **gazebo::physics::DARTBoxShape**

  *DART Box shape.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

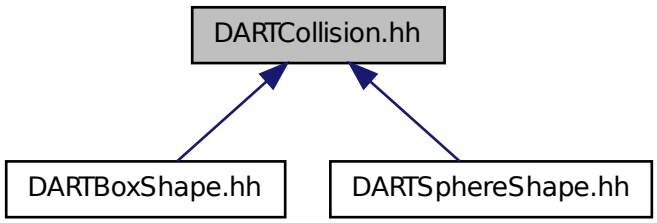
## 11.38 DARTCollision.hh File Reference

```
#include "gazebo/common/CommonTypes.hh"
```

```
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/dart/dart_inc.h"
```

Include dependency graph for DARTCollision.hh:

This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::DARTCollision**

  ***Base*** *(p. 159) class for all DART collisions.*

### Namespaces

- namespace **gazebo**

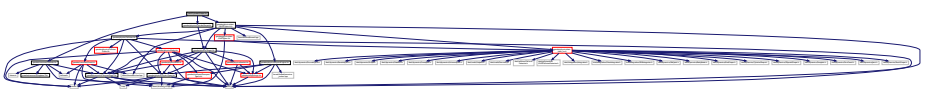
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.39 DARTCylinderShape.hh File Reference

```
#include "gazebo/common/Console.hh"
#include "gazebo/physics/CylinderShape.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
```

Include dependency graph for DARTCylinderShape.hh:



### Classes

- class **gazebo::physics::DARTCylinderShape**

  *DART cylinder shape.*

### Namespaces

- namespace **gazebo**

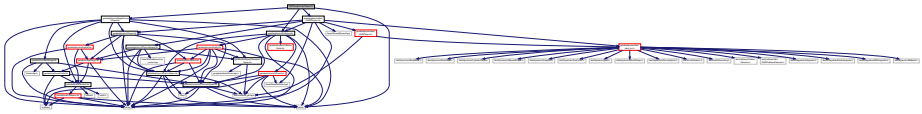
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.40 DARTHeightmapShape.hh File Reference

```
#include <vector>
#include "gazebo/physics/HeightmapShape.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/physics/Collision.hh"
```

Include dependency graph for DARTHeightmapShape.hh:



### Classes

- class **gazebo::physics::DARTHeightmapShape**

  *DART Height map collision.*

### Namespaces

- namespace **gazebo**

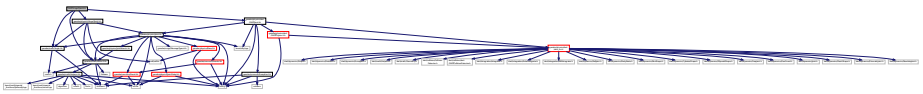
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.41 DARTHinge2Joint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Hinge2Joint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
```

Include dependency graph for DARTHinge2Joint.hh:



### Classes

- class **gazebo::physics::DARTHinge2Joint**

  *A two axis hinge joint.*

### Namespaces

- namespace **gazebo**

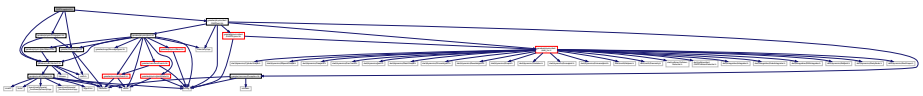
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.42 DARTHingeJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/HingeJoint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
```

Include dependency graph for DARTHingeJoint.hh:



### Classes

- class **gazebo::physics::DARTHingeJoint**

  *A single axis hinge joint.*

### Namespaces

- namespace **gazebo**

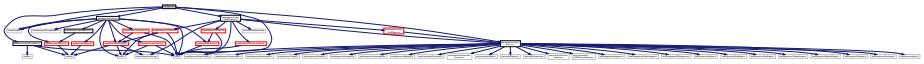
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

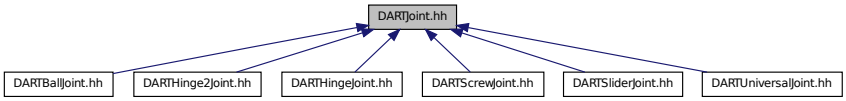
## 11.43 DARTJoint.hh File Reference

```
#include <boost/any.hpp>
#include <string>
#include "gazebo/common/Exception.hh"
#include "gazebo/physics/Joint.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTPhysics.hh"
```

Include dependency graph for DARTJoint.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::DARTJoint**

  *DART joint interface.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.44 DARTLink.hh File Reference

```
#include <vector>
#include "gazebo/physics/Link.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTTypes.hh"
```

Include dependency graph for DARTLink.hh:

### Classes

- class **gazebo::physics::DARTLink**

  *DART **Link** (p. 542) class.*

### Namespaces

- namespace **gazebo**

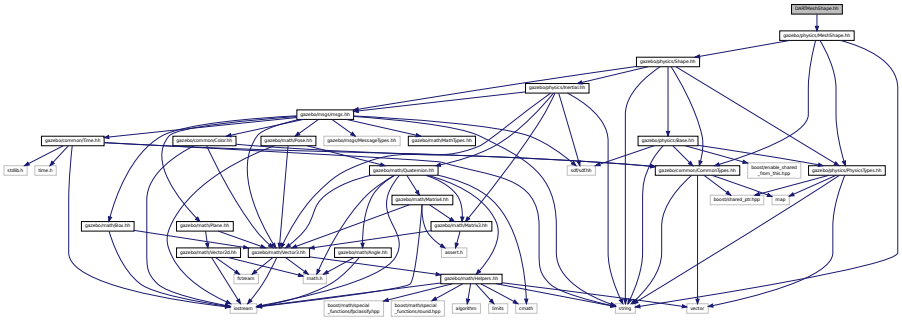
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.45 DARTMeshShape.hh File Reference

```
#include "gazebo/physics/MeshShape.hh"
```

Include dependency graph for DARTMeshShape.hh:



### Classes

- class **gazebo::physics::DARTMeshShape**

  *Triangle mesh collision.*

### Namespaces

- namespace **gazebo**

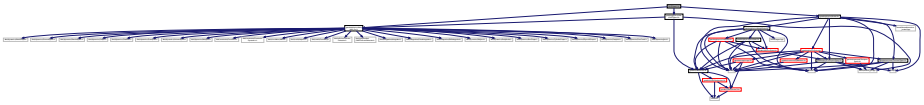
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.46 DARTModel.hh File Reference

```
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/physics/Model.hh"
```

Include dependency graph for DARTModel.hh:



### Classes

- class **gazebo::physics::DARTModel**

  *DART model class.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.47 DARTMultiRayShape.hh File Reference

```
#include "gazebo/physics/MultiRayShape.hh"
#include "gazebo/physics/dart/DARTTypes.hh"
```

Include dependency graph for DARTMultiRayShape.hh:

### Classes

- class **gazebo::physics::DARTMultiRayShape**

  *DART specific version of* ***MultiRayShape*** *(*p. *664).*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

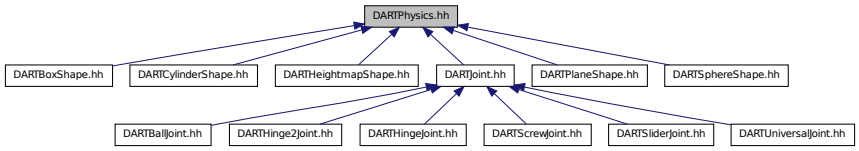
  *namespace for physics*

## 11.48 DARTPhysics.hh File Reference

```
#include <string>
#include <boost/thread/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "gazebo/physics/PhysicsEngine.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTTypes.hh"
```

Include dependency graph for DARTPhysics.hh:

This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::DARTPhysics**

  *DART physics engine.*

### Namespaces

- namespace **gazebo**

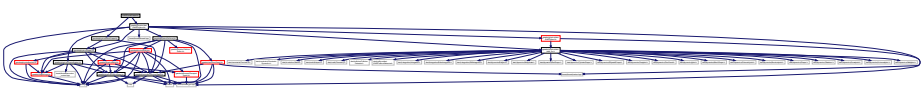
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.49 DARTPlaneShape.hh File Reference

```
#include "gazebo/physics/PlaneShape.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
```

Include dependency graph for DARTPlaneShape.hh:



### Classes

- class **gazebo::physics::DARTPlaneShape**

  *An DART Plane shape.*

### Namespaces

- namespace **gazebo**

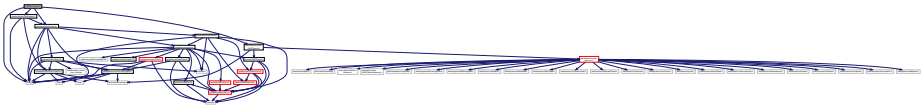
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.50 DARTRayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/RayShape.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/dart/DARTTypes.hh"
```

Include dependency graph for DARTRayShape.hh:



### Classes

- class **gazebo::physics::DARTRayShape**

  *Ray collision.*

### Namespaces

- namespace **gazebo**

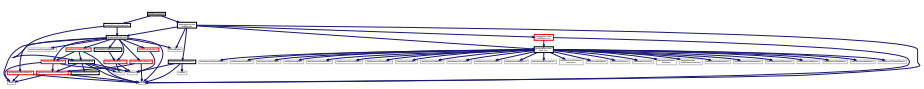
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.51 DARTScrewJoint.hh File Reference

```
#include "gazebo/physics/ScrewJoint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
```

Include dependency graph for DARTScrewJoint.hh:



### Classes

- class **gazebo::physics::DARTScrewJoint**

  *A screw joint.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.52 DARTSliderJoint.hh File Reference

```
#include "gazebo/physics/SliderJoint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
```

Include dependency graph for DARTSliderJoint.hh:

### Classes

- class **gazebo::physics::DARTSliderJoint**

  *A slider joint.*

### Namespaces

- namespace **gazebo**

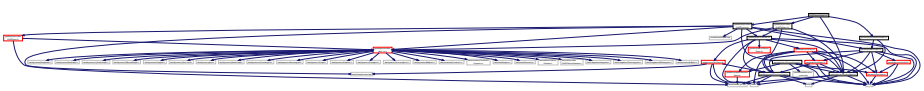
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.53 DARTSphereShape.hh File Reference

```
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/physics/dart/DARTCollision.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/SphereShape.hh"
```

Include dependency graph for DARTSphereShape.hh:



### Classes

- class **gazebo::physics::DARTSphereShape**

  *A DART sphere shape.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

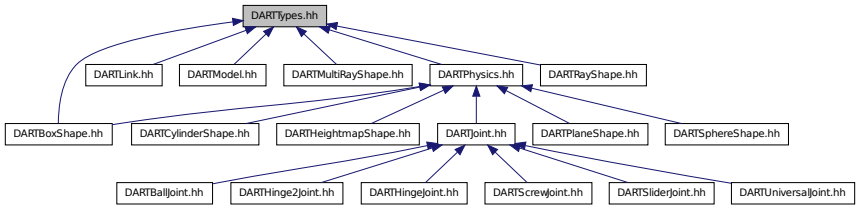
## 11.54 DARTTypes.hh File Reference

DART wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/dart/dart_inc.h"
```

Include dependency graph for DARTTypes.hh:

This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::DARTTypes**

  *A set of functions for converting between the math types used by gazebo and dart.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

### Typedefs

- typedef boost::shared_ptr < DARTCollision > **gazebo::physics::DARTCollisionPtr**
- typedef boost::shared_ptr < DARTJoint > **gazebo::physics::DARTJointPtr**
- typedef boost::shared_ptr < DARTLink > **gazebo::physics::DARTLinkPtr**
- typedef boost::shared_ptr < DARTModel > **gazebo::physics::DARTModelPtr**
- typedef boost::shared_ptr < DARTPhysics > **gazebo::physics::DARTPhysicsPtr**
- typedef boost::shared_ptr < DARTRayShape > **gazebo::physics::DARTRayShapePtr**

#### 11.54.1 Detailed Description

DART wrapper forward declarations and typedefs.

## 11.55 DARTUniversalJoint.hh File Reference

```
#include "gazebo/physics/UniversalJoint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
```

Include dependency graph for DARTUniversalJoint.hh:

### Classes

- class **gazebo::physics::DARTUniversalJoint**

  *A universal joint.*

### Namespaces

- namespace **gazebo**

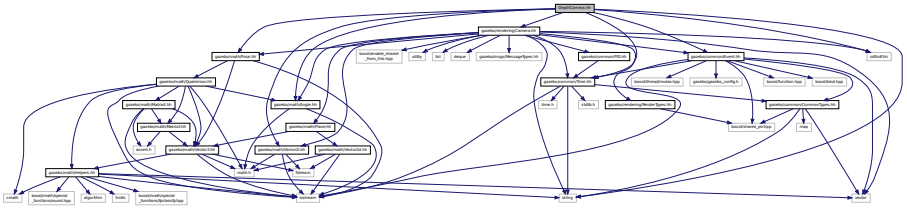
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.56 DepthCamera.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Vector2i.hh"
#include "gazebo/rendering/Camera.hh"
```

Include dependency graph for DepthCamera.hh:



### Classes

- class **gazebo::rendering::DepthCamera**

  *Depth camera used to render depth data into an image buffer.*

### Namespaces

- namespace **gazebo**

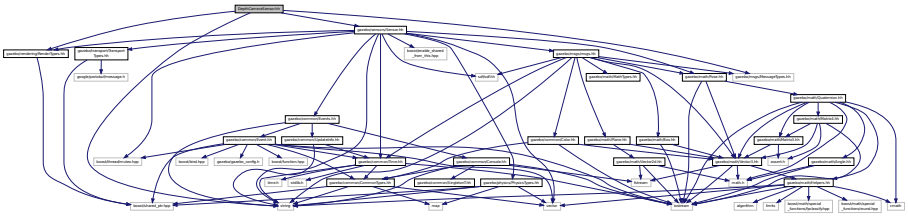
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*
- namespace **Ogre**

## 11.57 DepthCameraSensor.hh File Reference

```
#include <string>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for DepthCameraSensor.hh:



### Classes

- class **gazebo::sensors::DepthCameraSensor**

### Namespaces

- namespace **gazebo**

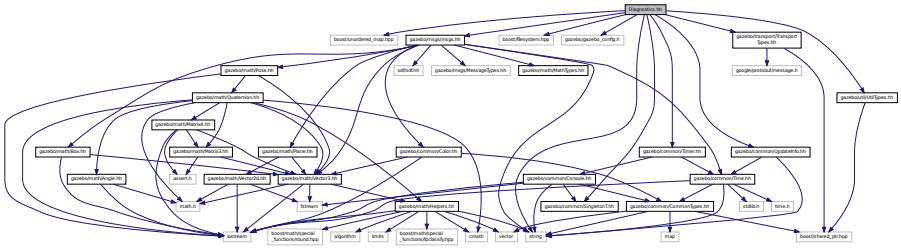
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.58 Diagnostics.hh File Reference

```
#include <boost/unordered_map.hpp>
#include <string>
#include <boost/filesystem.hpp>
#include "gazebo/gazebo_config.h"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/Timer.hh"
#include "gazebo/util/UtilTypes.hh"
```

Include dependency graph for Diagnostics.hh:



### Classes

- class **gazebo::util::DiagnosticManager**

  *A diagnostic manager class.*
- class **gazebo::util::DiagnosticTimer**

  *A timer designed for diagnostics.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::util**

### Macros

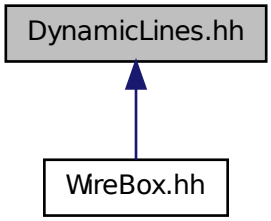
- #define **DIAG_TIMER_LAP**(_name, _prefix) ((void)0)
- #define **DIAG_TIMER_START**(_name) ((void) 0)
- #define **DIAG_TIMER_STOP**(_name) ((void) 0)

## 11.59 DynamicLines.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/common/CommonIface.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/rendering/Conversions.hh"
#include "gazebo/rendering/DynamicRenderable.hh"
```

Include dependency graph for DynamicLines.hh:

This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::rendering::DynamicLines**

  *Class for drawing lines that can change.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

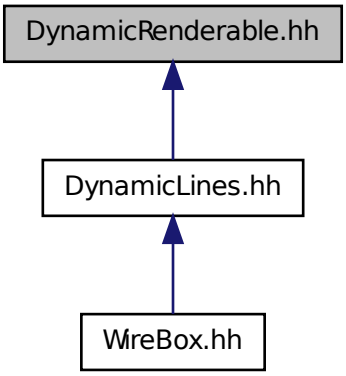
  *Rendering namespace.*

## 11.60 DynamicRenderable.hh File Reference

```
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for DynamicRenderable.hh:

This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::rendering::DynamicRenderable**

  *Abstract base class providing mechanisms for dynamically growing hardware buffers.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

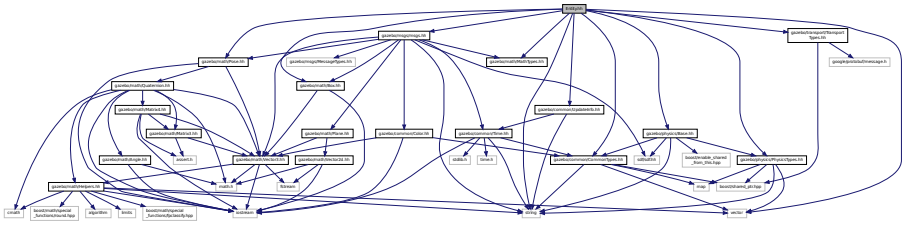
  *Rendering namespace.*

## 11.61 Entity.hh File Reference

```
#include <string>
```

```
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Entity.hh:



This graph shows which files directly or indirectly include this file:

### Classes

- class **gazebo::physics::Entity**

  ***Base*** *(p. 159) class for all physics objects in Gazebo.*

### Namespaces

- namespace **boost**
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

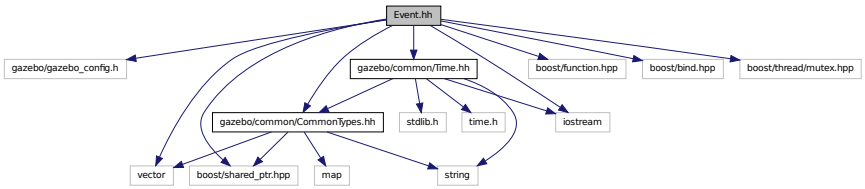
  *namespace for physics*

## 11.62 Event.hh File Reference

```
#include <gazebo/gazebo_config.h>
```

```
#include <gazebo/common/Time.hh>
#include <gazebo/common/CommonTypes.hh>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <iostream>
#include <vector>
```

Include dependency graph for Event.hh:




This graph shows which files directly or indirectly include this file:

## Classes

- class **gazebo::event::Connection**

  *A class that encapsulates a connection.*
- class **gazebo::event::Event**

  *Base class for all events.*
- class **gazebo::event::EventT< T >**

  *A class for event processing.*

## Namespaces

- namespace **gazebo**

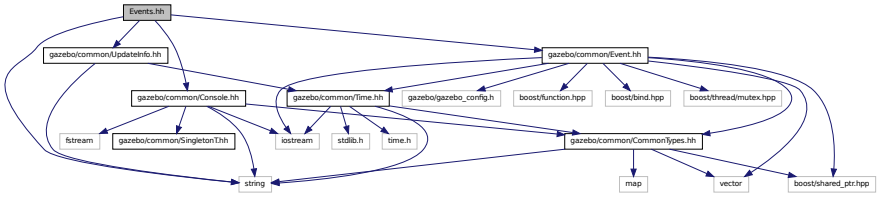
  *Forward declarations for the common classes.*
- namespace **gazebo::event**

  ***Event*** *(p. 390) namespace.*

# 11.63 Events.hh File Reference

```
#include <string>
#include "gazebo/common/Console.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
```

Include dependency graph for Events.hh:




This graph shows which files directly or indirectly include this file:

## Classes

- class **gazebo::event::Events**

  *An **Event** (p. 390) class to get notifications for simulator events.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

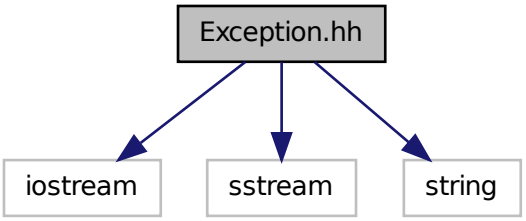
- namespace **gazebo::event**

  ***Event** (p. 390) namespace.*

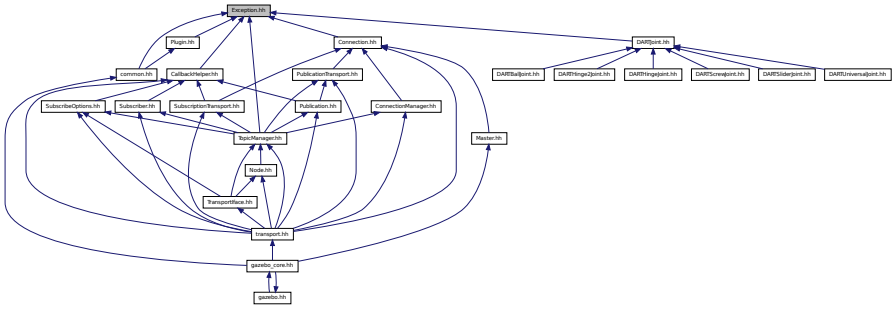
# 11.64 Exception.hh File Reference

```
#include <iostream>
#include <sstream>
#include <string>
```

Include dependency graph for Exception.hh:




This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::common::AssertionInternalError**

  *Class for generating Exceptions which come from gazebo assertions.*

- class **gazebo::common::Exception**

  *Class for generating exceptions.*

- class **gazebo::common::InternalError**

  *Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::common**

  *Common namespace.*

### Macros

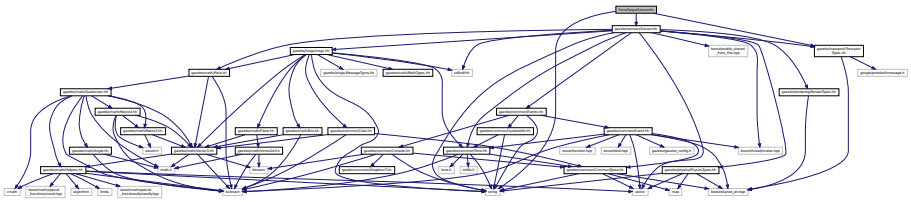
- #define **gzthrow**(msg)

  *This macro logs an error to the throw stream and throws an exception that contains the file name and line number.*

## 11.65 ForceTorqueSensor.hh File Reference

```
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for ForceTorqueSensor.hh:



### Classes

- class **gazebo::sensors::ForceTorqueSensor**

  ***Sensor*** (p. *837) for measure force and torque on a joint.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

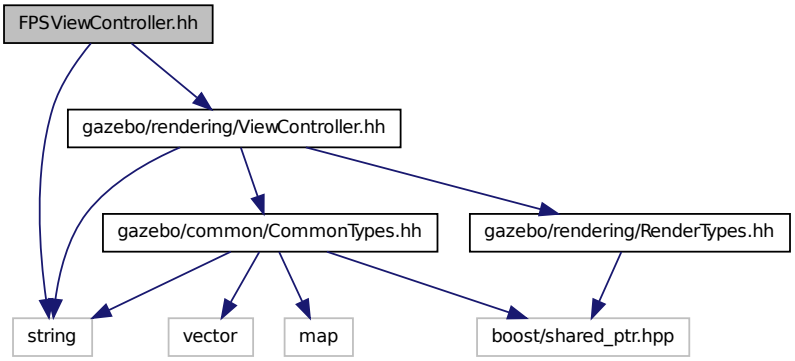
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.66 FPSViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/ViewController.hh"
```

Include dependency graph for FPSViewController.hh:




### Classes

- class **gazebo::rendering::FPSViewController**

  *First Person Shooter style view controller.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

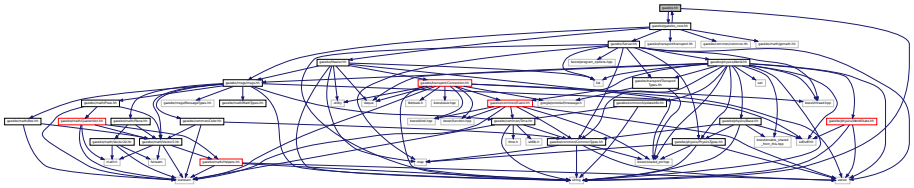
- namespace **gazebo::rendering**

  *Rendering namespace.*

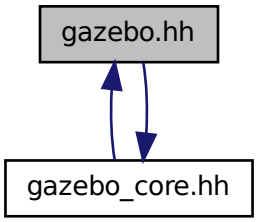
## 11.67 gazebo.hh File Reference

```
#include <gazebo/gazebo_core.hh>
#include <string>
```

Include dependency graph for gazebo.hh:



This graph shows which files directly or indirectly include this file:




### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

### Functions

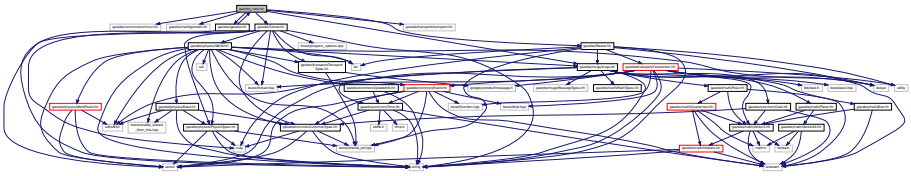
- void **gazebo::add_plugin** (const std::string &_filename)
- std::string **gazebo::find_file** (const std::string &_file)

  *Find a file in the gazebo search paths.*
- void **gazebo::fini** ()
- bool **gazebo::init** ()
- bool **gazebo::load** (int _argc=0, char **_argv=0)
- void **gazebo::print_version** ()
- void **gazebo::run** ()
- void **gazebo::stop** ()

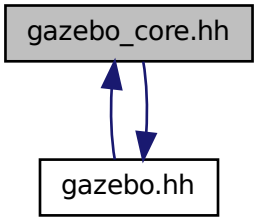
## 11.68 gazebo_core.hh File Reference

```
#include <gazebo/common/common.hh>
#include <gazebo/math/gzmath.hh>
#include <gazebo/msgs/msgs.hh>
#include <gazebo/transport/transport.hh>
#include <gazebo/Server.hh>
#include <gazebo/Master.hh>
#include <gazebo/gazebo.hh>
```

Include dependency graph for gazebo_core.hh:



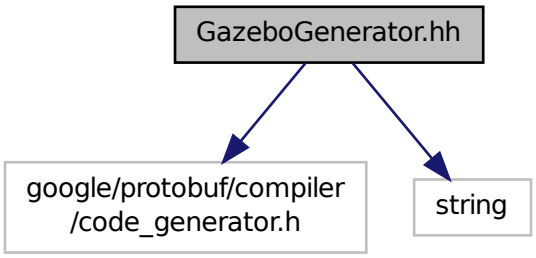
This graph shows which files directly or indirectly include this file:




## 11.69 GazeboGenerator.hh File Reference

```
#include <google/protobuf/compiler/code_generator.h>
#include <string>
```

Include dependency graph for GazeboGenerator.hh:



### Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**

  *Google protobuf message generator for **gazebo::msgs*** (p. *108*).

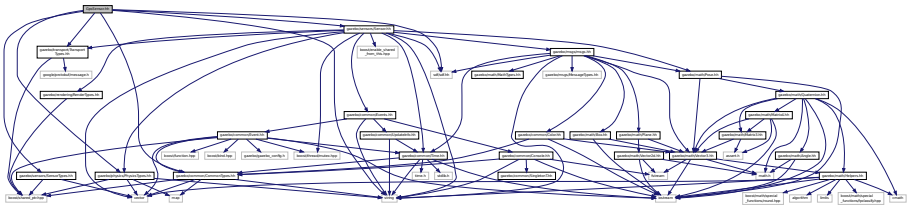
### Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

## 11.70 GpsSensor.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for GpsSensor.hh:



### Classes

- class **gazebo::sensors::GpsSensor**

  ***GpsSensor*** (p. *426*) *to provide position measurement.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.71 GpuLaser.hh File Reference

```
#include <string>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/rendering/Camera.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Vector2i.hh"
```

Include dependency graph for GpuLaser.hh:

### Classes

- class **gazebo::rendering::GpuLaser**

  *GPU based laser distance sensor.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::common**

  *Common namespace.*

- namespace **gazebo::rendering**

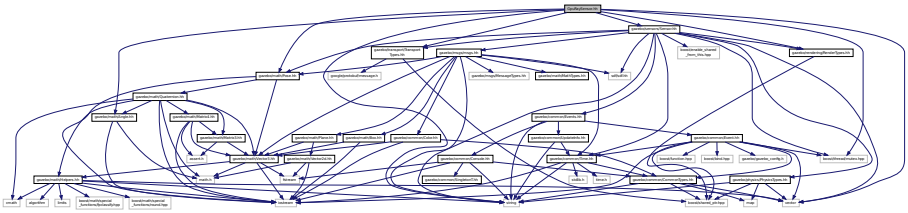
  *Rendering namespace.*

- namespace **Ogre**

## 11.72 GpuRaySensor.hh File Reference

```
#include <vector>
#include <string>
#include <boost/thread/mutex.hpp>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for GpuRaySensor.hh:



### Classes

- class **gazebo::sensors::GpuRaySensor**

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.73 Grid.hh File Reference

```
#include <stdint.h>
#include <vector>
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Color.hh"
```

Include dependency graph for Grid.hh:

### Classes

- class **gazebo::rendering::Grid**

  *Displays a grid of cells, drawn with lines.*

### Namespaces

- namespace **gazebo**

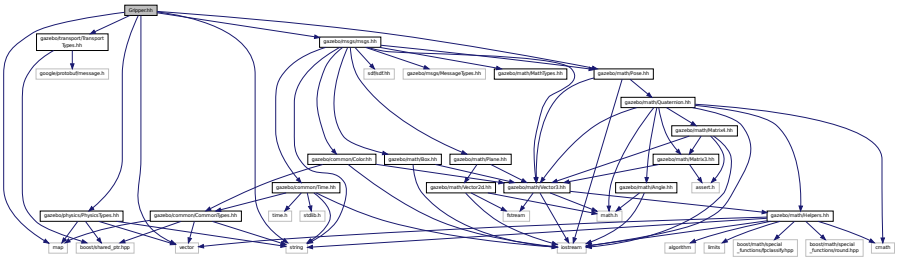
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*
- namespace **Ogre**

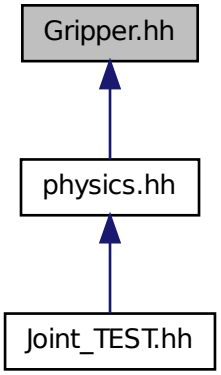
## 11.74 Gripper.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for Gripper.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::Gripper**
  *A gripper abstraction.*

### Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

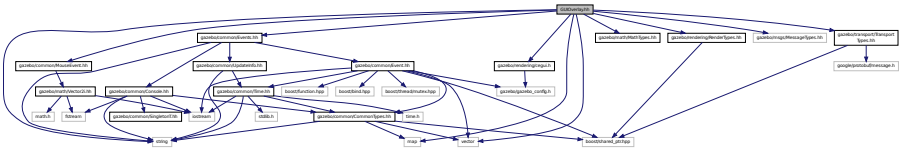
- namespace **gazebo::physics**

  *namespace for physics*

## 11.75 GUIOverlay.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/rendering/cegui.h"
#include "gazebo/common/MouseEvent.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for GUIOverlay.hh:



### Classes

- class **gazebo::rendering::GUIOverlay**

  *A class that creates a CEGUI overlay on a render window.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::rendering**

  *Rendering namespace.*

- namespace **Ogre**

## 11.76 Heightmap.hh File Reference

```
#include <string>
#include <vector>
#include <boost/filesystem.hpp>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/rendering/Scene.hh"
```

Include dependency graph for Heightmap.hh:

### Classes

- class **gazebo::rendering::DummyPageProvider**

  *Pretends to provide procedural page content to avoid page loading.*
- class **gazebo::rendering::GzTerrainMatGen**
- class **gazebo::rendering::Heightmap**

  *Rendering a terrain using heightmap information.*
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg**

  *Keeping the CG shader for reference.*
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL**

  *Utility class to help with generating shaders for GLSL.*
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile**

  *Shader model 2 profile target.*

### Namespaces

- namespace **gazebo**

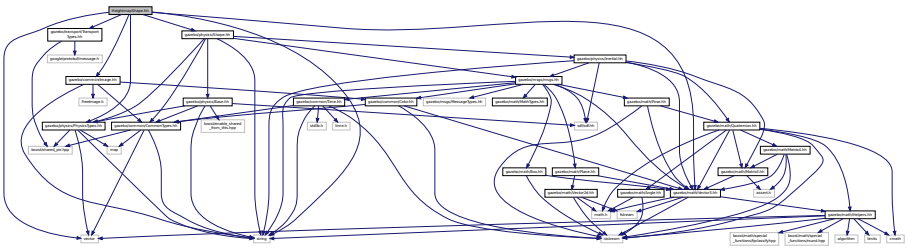
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*
- namespace **Ogre**

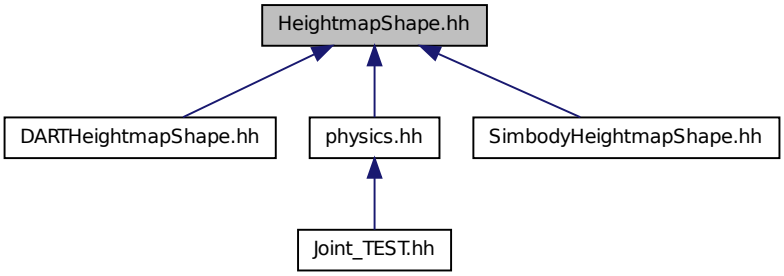
## 11.77 HeightmapShape.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for HeightmapShape.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::HeightmapShape**

  ***HeightmapShape*** *(p. 465) collision shape builds a heightmap from an image.*

### Namespaces

- namespace **gazebo**

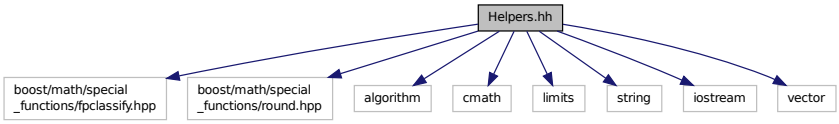
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

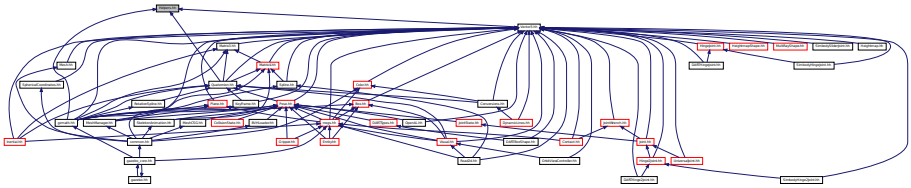
## 11.78 Helpers.hh File Reference

```
#include <boost/math/special_functions/fpclassify.hpp>
#include <boost/math/special_functions/round.hpp>
#include <algorithm>
#include <cmath>
#include <limits>
#include <string>
#include <iostream>
#include <vector>
```

Include dependency graph for Helpers.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

## Macros

- #define **GZ_DBL_MAX** std::numeric_limits<double>::max()

  *Double maximum value.*
- #define **GZ_DBL_MIN** std::numeric_limits<double>::min()

  *Double min value.*
- #define **GZ_FLT_MAX** std::numeric_limits<float>::max()

  *Float maximum value.*
- #define **GZ_FLT_MIN** std::numeric_limits<float>::min()

  *Float minimum value.*
- #define **GZ_UINT32_MAX** std::numeric_limits<uint32_t>::max()

  *32bit unsigned integer maximum value*
- #define **GZ_UINT32_MIN** std::numeric_limits<uint32_t>::min()

  *32bit unsigned integer minimum value*

## Functions

- template<typename T >
  T **gazebo::math::clamp** (T _v, T _min, T _max)

  *Simple clamping function.*
- template<typename T >
  bool **gazebo::math::equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

  *check if two values are equal, within a tolerance*
- float **gazebo::math::fixnan** (float _v)

  *Fix a nan value.*
- double **gazebo::math::fixnan** (double _v)

  *Fix a nan value.*
- bool **gazebo::math::isnan** (float _v)

  *check if a float is NaN*
- bool **gazebo::math::isnan** (double _v)

*check if a double is NaN*

- bool **gazebo::math::isPowerOfTwo** (unsigned int _x)

  *is this a power of 2?*

- template<typename T >
  T **gazebo::math::max** (const std::vector< T > &_values)

  *get the maximum value of vector of values*

- template<typename T >
  T **gazebo::math::mean** (const std::vector< T > &_values)

  *get mean of vector of values*

- template<typename T >
  T **gazebo::math::min** (const std::vector< T > &_values)

  *get the minimum value of vector of values*

- double **gazebo::math::parseFloat** (const std::string &_input)

  *parse string into float*

- int **gazebo::math::parseInt** (const std::string &_input)

  *parse string into an integer*

- template<typename T >
  T **gazebo::math::precision** (const T &_a, const unsigned int &_precision)

  *get value at a specified precision*

- template<typename T >
  T **gazebo::math::variance** (const std::vector< T > &_values)

  *get variance of vector of values*

## Variables

- static const double **gazebo::math::NAN_D** = std::numeric_limits<double>::quiet_NaN()

  *Returns the representation of a quiet not a number (NAN)*

- static const int **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()

  *Returns the representation of a quiet not a number (NAN)*

### 11.78.1 Macro Definition Documentation

#### 11.78.1.1 #define GZ_DBL_MAX std::numeric_limits<double>::max()

Double maximum value.

#### 11.78.1.2 #define GZ_DBL_MIN std::numeric_limits<double>::min()

Double min value.

#### 11.78.1.3 #define GZ_FLT_MAX std::numeric_limits<float>::max()

Float maximum value.

#### 11.78.1.4 #define GZ_FLT_MIN std::numeric_limits<float>::min()

Float minimum value.

#### 11.78.1.5 #define GZ_UINT32_MAX std::numeric_limits<uint32_t>::max()

32bit unsigned integer maximum value

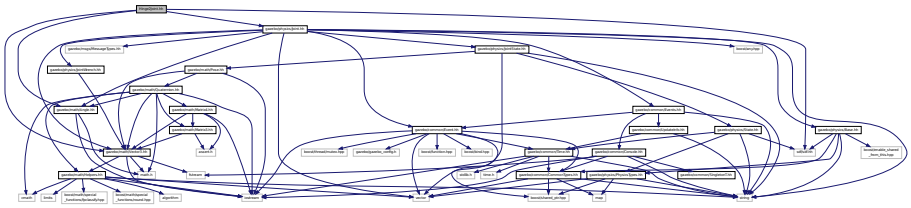
#### 11.78.1.6 #define GZ_UINT32_MIN std::numeric_limits<uint32_t>::min()

32bit unsigned integer minimum value

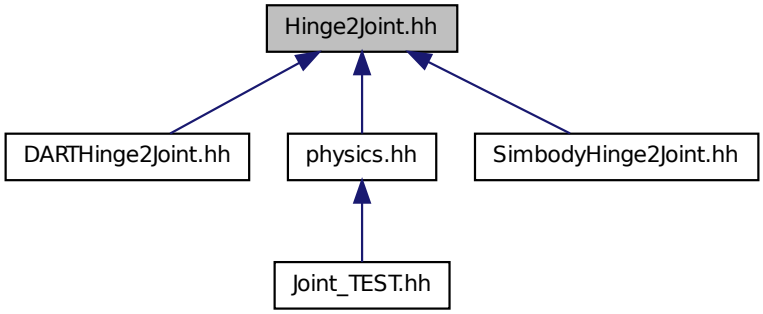
## 11.79 Hinge2Joint.hh File Reference

```
#include <sdf/sdf.hh>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for Hinge2Joint.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::Hinge2Joint< T >**

  *A two axis hinge joint.*

### Namespaces

- namespace **gazebo**

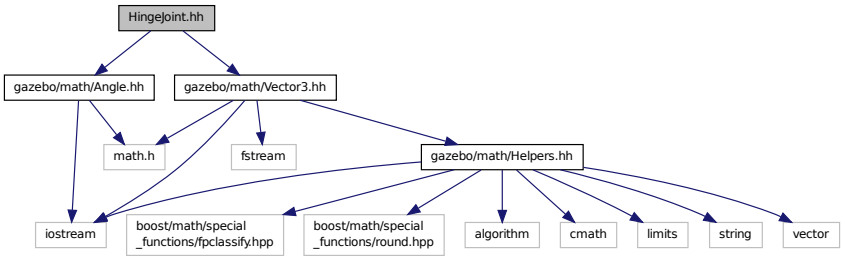
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

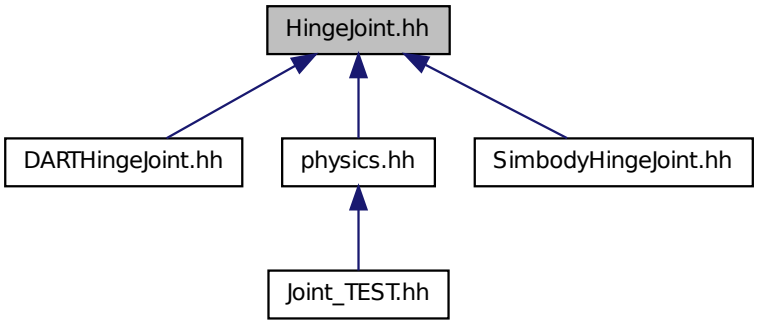
## 11.80 HingeJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for HingeJoint.hh:




This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::HingeJoint**< **T** >

  *A single axis hinge joint.*

### Namespaces

- namespace **gazebo**

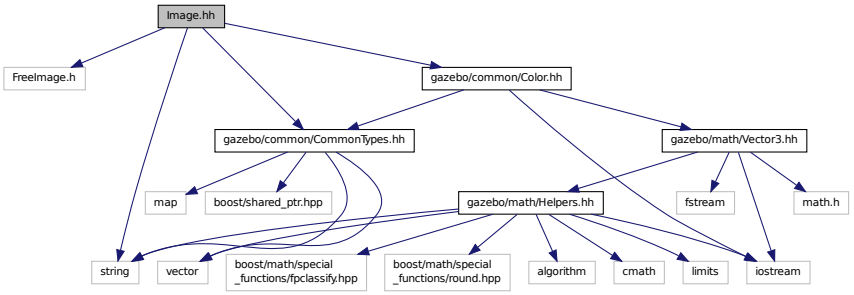
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

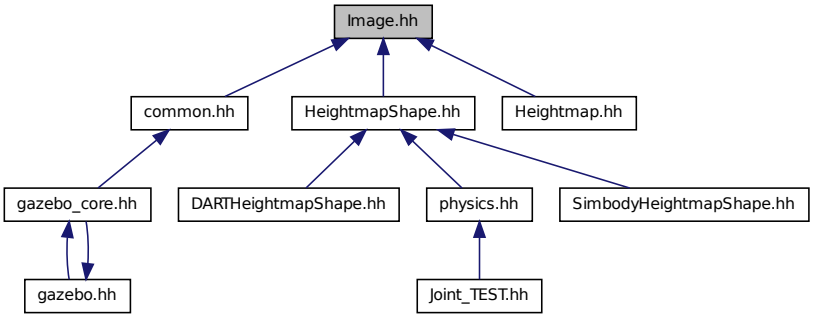
## 11.81 Image.hh File Reference

```
#include <FreeImage.h>
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/Color.hh"
```

Include dependency graph for Image.hh:




This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::Image**

  *Encapsulates an image.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

### Variables

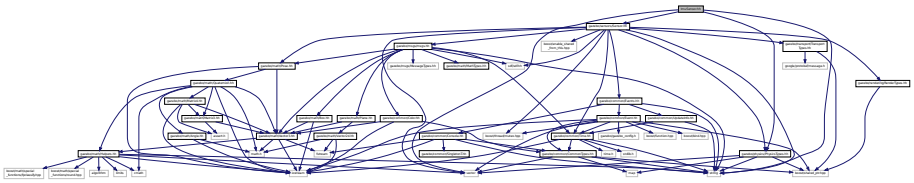
- static std::string **gazebo::common::PixelFormatNames** [ ]

  *String names for the pixel formats.*

## 11.82 ImuSensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for ImuSensor.hh:



### Classes

- class **gazebo::sensors::ImuSensor**

  *An IMU sensor.*

### Namespaces

- namespace **gazebo**

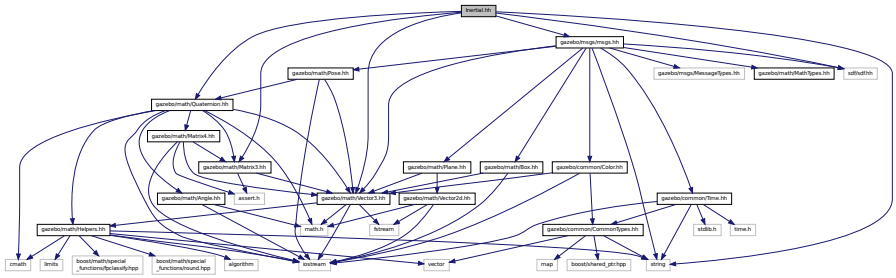
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

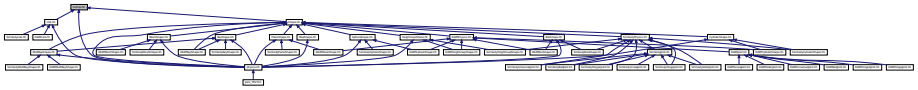
## 11.83 Inertial.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
```

Include dependency graph for Inertial.hh:




This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::physics::Inertial**

  *A class for inertial information about a link.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

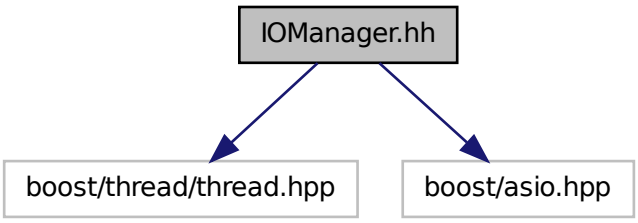
- namespace **gazebo::physics**

  *namespace for physics*

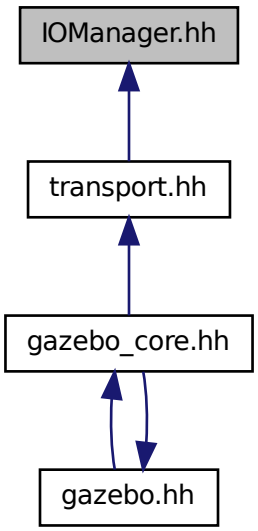
# 11.84 IOManager.hh File Reference

```
#include <boost/thread/thread.hpp>
#include <boost/asio.hpp>
```

Include dependency graph for IOManager.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::IOManager**

  *Manages boost::asio IO.*

## Namespaces

- namespace **gazebo**

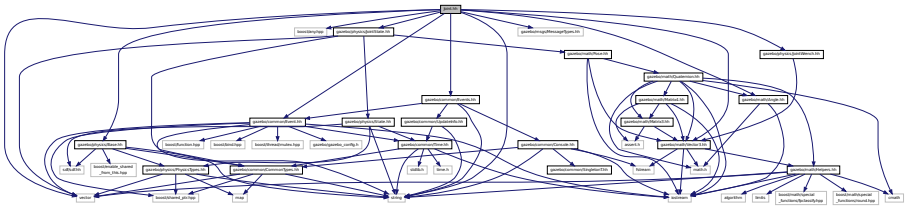
*Forward declarations for the common classes.*

- namespace **gazebo::transport**

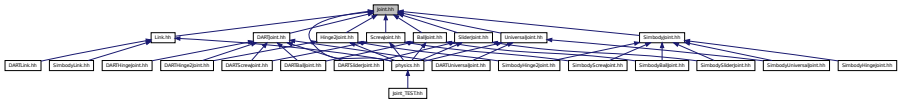
## 11.85 Joint.hh File Reference

```
#include <string>
#include <vector>
#include <boost/any.hpp>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/JointWrench.hh"
```

Include dependency graph for Joint.hh:



This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::physics::Joint**

  ***Base*** *(*p. *159) class for all joints.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::physics**

  *namespace for physics*

### Macros

- #define **MAX_JOINT_AXIS** 2

  *maximum number of axis per joint anticipated.*

### 11.85.1 Macro Definition Documentation

#### 11.85.1.1 #define MAX_JOINT_AXIS 2

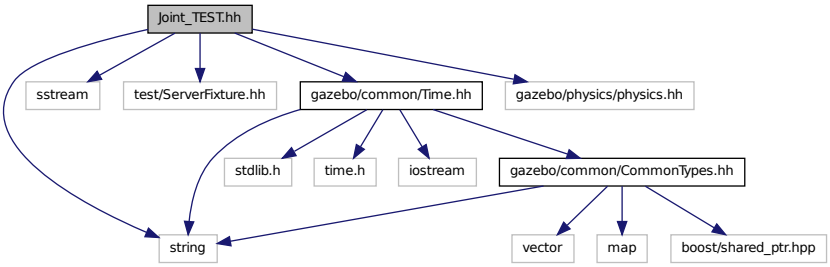
maximum number of axis per joint anticipated.

Currently, this is 2 as 3-axis joints (e.g. ball) actuation, control is not there yet.

## 11.86 Joint_TEST.hh File Reference

```
#include <string>
#include <sstream>
#include "test/ServerFixture.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/physics.hh"
```

Include dependency graph for Joint_TEST.hh:




### Classes

- class **Joint_TEST**
- class **Joint_TEST::SpawnJointOptions**

  *Class to hold parameters for spawning joints.*

### Typedefs

- typedef std::tr1::tuple< const char *, const char * > **std_string2**

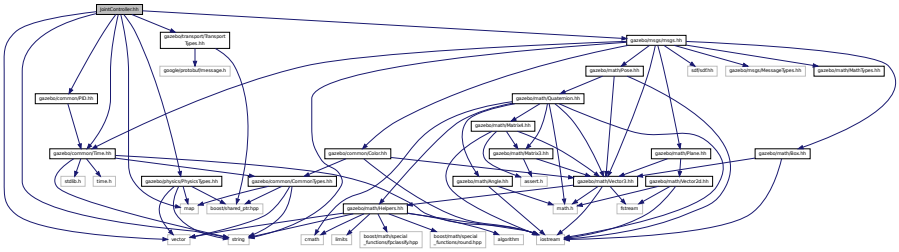
### 11.86.1 Typedef Documentation

#### 11.86.1.1 typedef std::tr1::tuple<const char *, const char *> std_string2

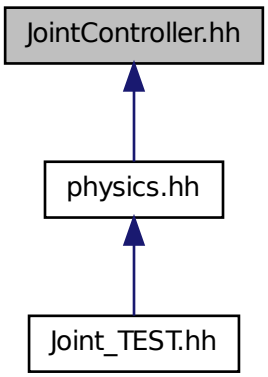
## 11.87 JointController.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
```

Include dependency graph for JointController.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::JointController**

*A class for manipulating **physics::Joint** (p. 496).*

### Namespaces

- namespace **gazebo**

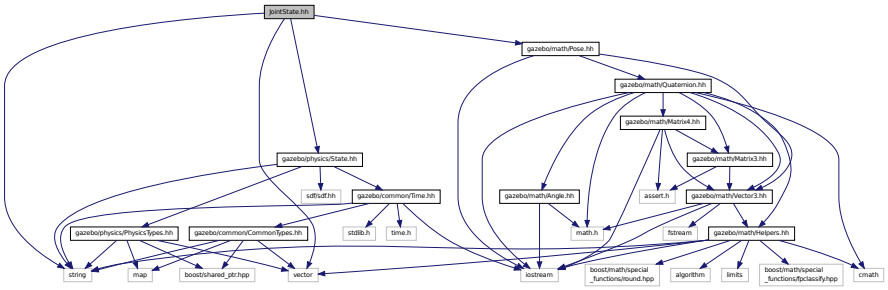
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

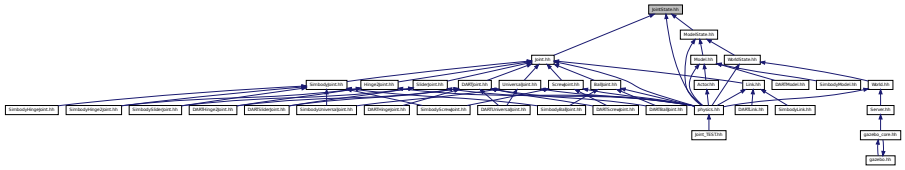
## 11.88 JointState.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.hh"
```

Include dependency graph for JointState.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::physics::JointState**

  *keeps track of state of a **physics::Joint** (p. 496)*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

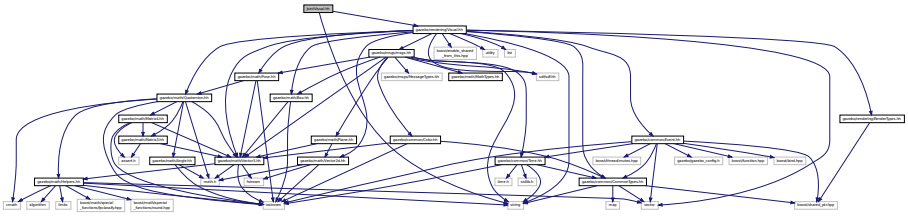
- namespace **gazebo::physics**

  *namespace for physics*

## 11.89 JointVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for JointVisual.hh:



### Classes

- class **gazebo::rendering::JointVisual**

  *Visualization for joints.*

### Namespaces

- namespace **gazebo**

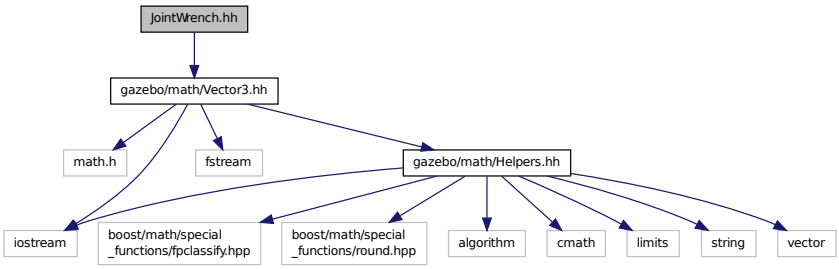
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

## 11.90 JointWrench.hh File Reference

```
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for JointWrench.hh:



This graph shows which files directly or indirectly include this file:

### Classes

- class **gazebo::physics::JointWrench**

  *Wrench information from a joint.*

### Namespaces

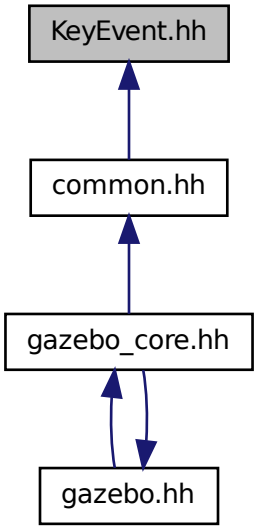
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.91 KeyEvent.hh File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::KeyEvent**

  *Generic description of a keyboard event.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

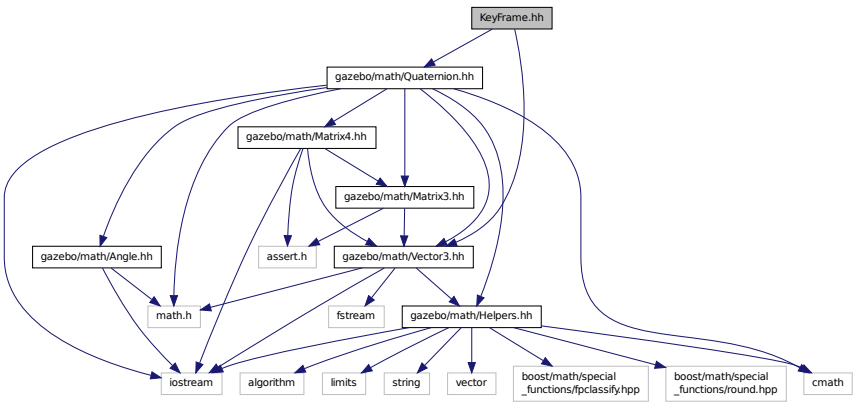
- namespace **gazebo::common**

  *Common namespace.*

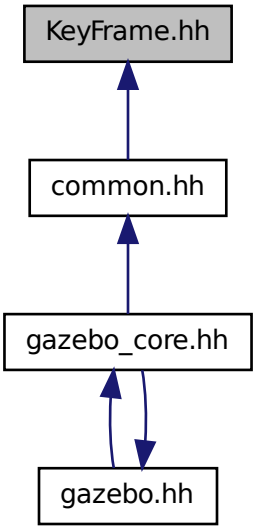
## 11.92 KeyFrame.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh"
```

Include dependency graph for KeyFrame.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::KeyFrame**

  *A key frame in an animation.*
- class **gazebo::common::NumericKeyFrame**

  *A keyframe for a **NumericAnimation*** (p. *692*).
- class **gazebo::common::PoseKeyFrame**

  *A keyframe for a **PoseAnimation*** (p. *743*).

### Namespaces

- namespace **gazebo**

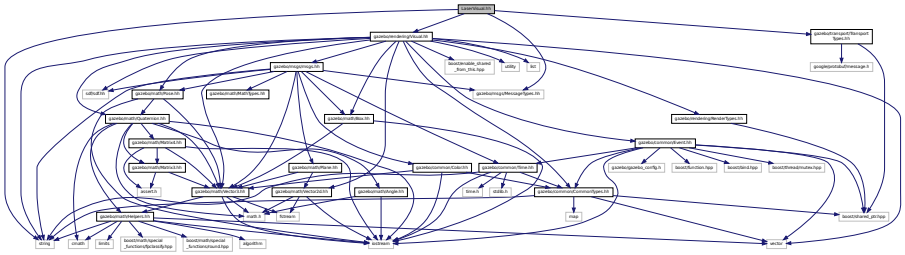
  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## 11.93 LaserVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for LaserVisual.hh:



### Classes

- class **gazebo::rendering::LaserVisual**

  *Visualization for laser data.*

### Namespaces

- namespace **gazebo**

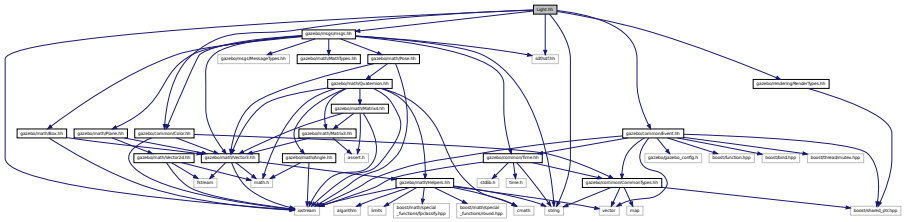
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

## 11.94 Light.hh File Reference

```
#include <string>
#include <iostream>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/Color.hh"
```

Include dependency graph for Light.hh:



### Classes

- class **gazebo::rendering::Light**

  *A light source.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::rendering**

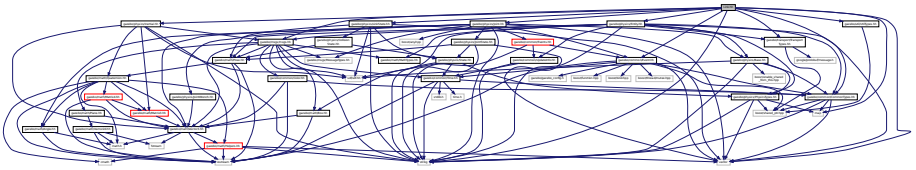
  *Rendering namespace.*

- namespace **Ogre**

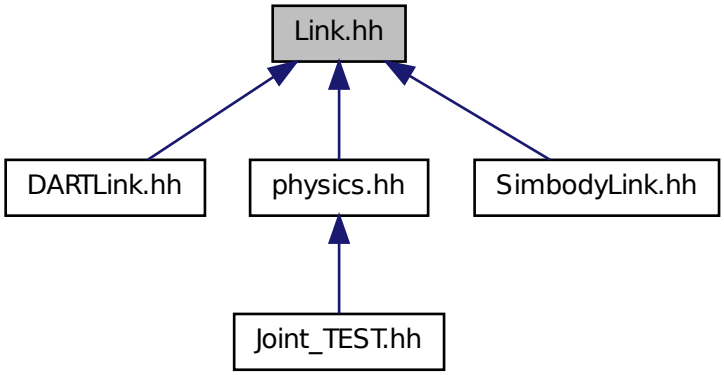
## 11.95 Link.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/UtilTypes.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/LinkState.hh"
#include "gazebo/physics/Entity.hh"
#include "gazebo/physics/Inertial.hh"
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for Link.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::Link**

  ***Link*** (p. *542*) *class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::physics**

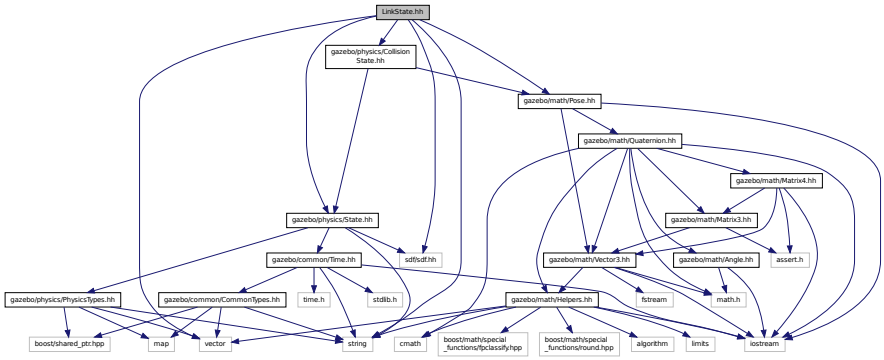
  *namespace for physics*

- namespace **gazebo::util**

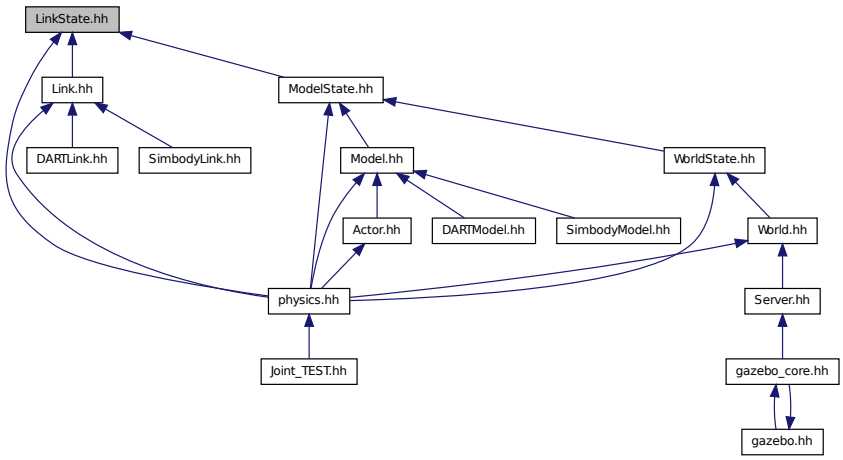
## 11.96 LinkState.hh File Reference

```
#include <vector>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/State.hh"
#include "gazebo/physics/CollisionState.hh"
#include "gazebo/math/Pose.hh"
```

Include dependency graph for LinkState.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::physics::LinkState**

  *Store state information of a **physics::Link** (p. 542) object.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

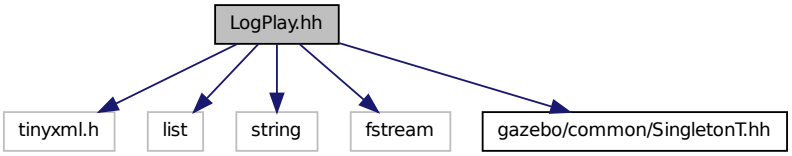
- namespace **gazebo::physics**

  *namespace for physics*

## 11.97 LogPlay.hh File Reference

```
#include <tinyxml.h>
#include <list>
#include <string>
#include <fstream>
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for LogPlay.hh:




### Classes

- class **gazebo::util::LogPlay**

### Namespaces

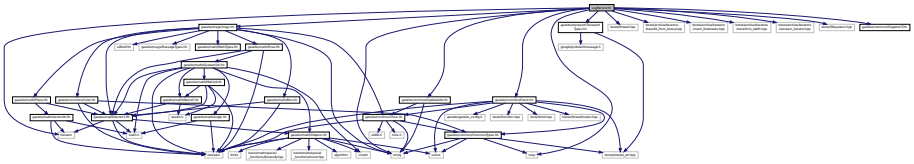
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::util**

## 11.98 LogRecord.hh File Reference

```
#include <fstream>
#include <string>
#include <map>
#include <boost/thread.hpp>
#include <boost/archive/iterators/base64_from_binary.hpp>
#include <boost/archive/iterators/insert_linebreaks.hpp>
#include <boost/archive/iterators/transform_width.hpp>
#include <boost/archive/iterators/ostream_iterator.hpp>
#include <boost/filesystem.hpp>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for LogRecord.hh:



### Classes

- class **gazebo::util::LogRecord**

  *addtogroup gazebo_util*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::util**

### Macros

- #define **GZ_LOG_VERSION** "1.0"

#### 11.98.1 Macro Definition Documentation

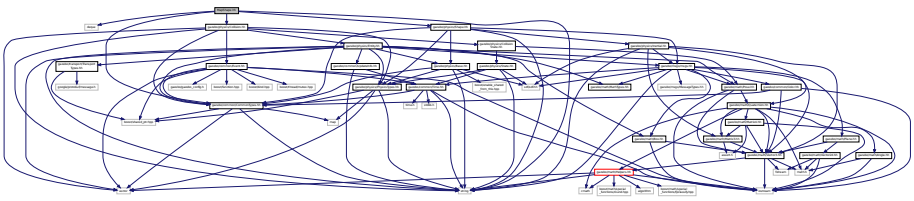
##### 11.98.1.1 #define GZ_LOG_VERSION "1.0"

## 11.99 mainpage.html File Reference

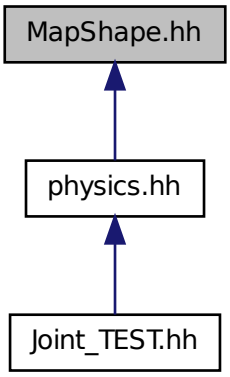
## 11.100 MapShape.hh File Reference

```
#include <deque>
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for MapShape.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::MapShape**

  *Creates box extrusions based on an image.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

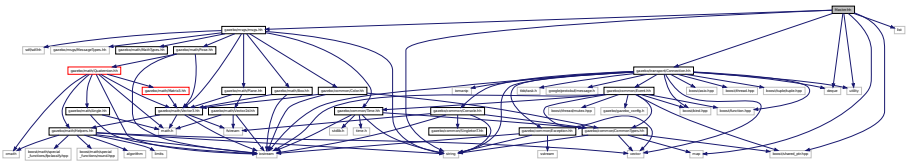
- namespace **gazebo::physics**

  *namespace for physics*

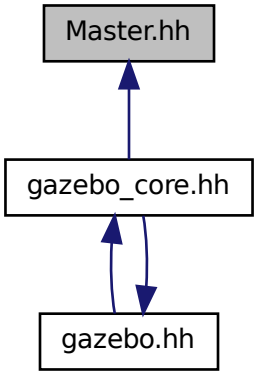
## 11.101 Master.hh File Reference

```
#include <string>
#include <list>
#include <deque>
#include <utility>
#include <map>
#include <boost/shared_ptr.hpp>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/Connection.hh"
```

Include dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::Master**

  *A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.*

### Namespaces

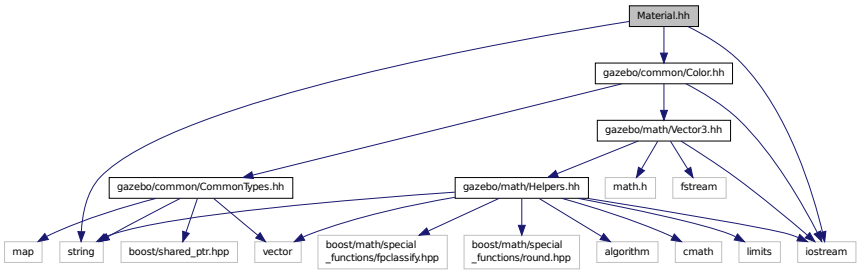
- namespace **gazebo**

  *Forward declarations for the common classes.*

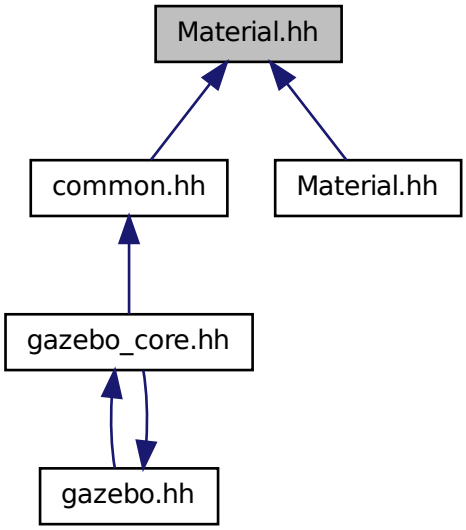
## 11.102 Material.hh File Reference

```
#include <string>
#include <iostream>
#include "gazebo/common/Color.hh"
```

Include dependency graph for common/Material.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::common::Material**

  *Encapsulates description of a material.*

## Namespaces

- namespace **gazebo**

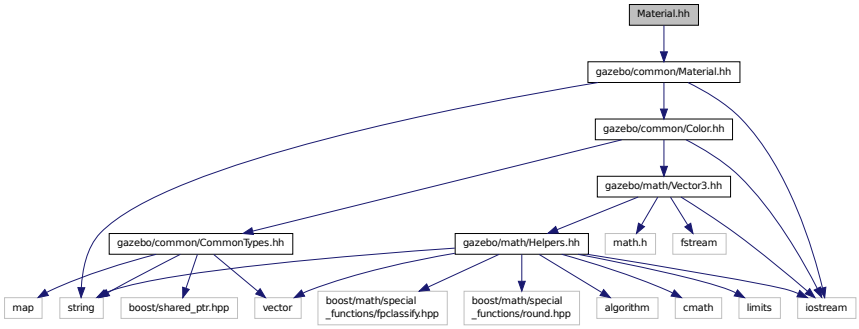
  *Forward declarations for the common classes.*
- namespace **gazebo::common**

*Common namespace.*

## 11.103 Material.hh File Reference

```
#include "gazebo/common/Material.hh"
```

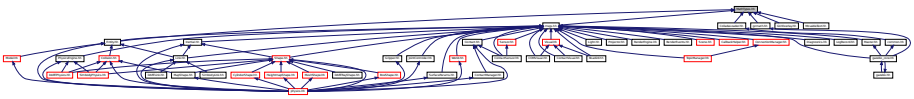
Include dependency graph for rendering/Material.hh:




## 11.104 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

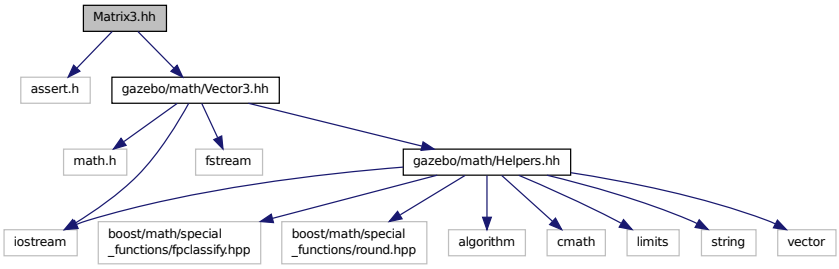
### 11.104.1 Detailed Description

Forward declarations for the math classes.

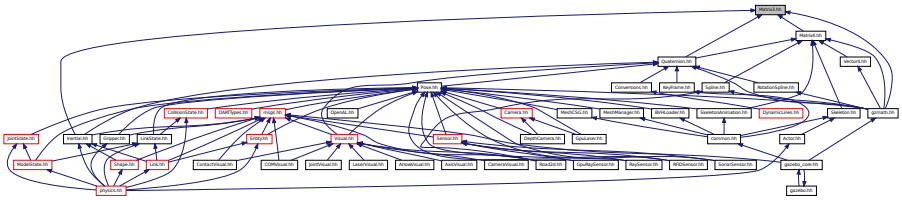
## 11.105 Matrix3.hh File Reference

```
#include <assert.h>
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for Matrix3.hh:




This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::math::Matrix3**

  *A 3x3 matrix class.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

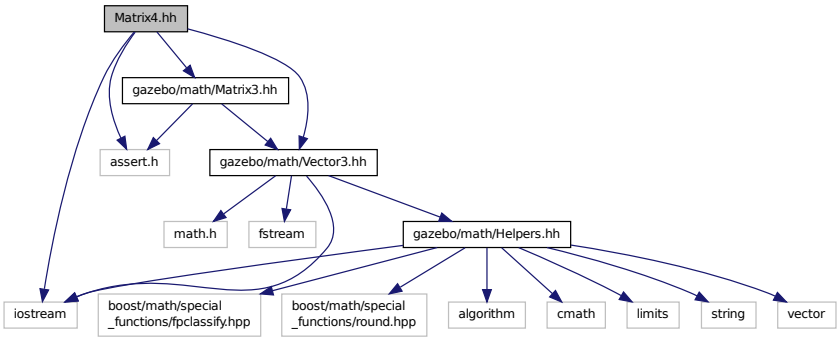
- namespace **gazebo::math**

  *Math namespace.*

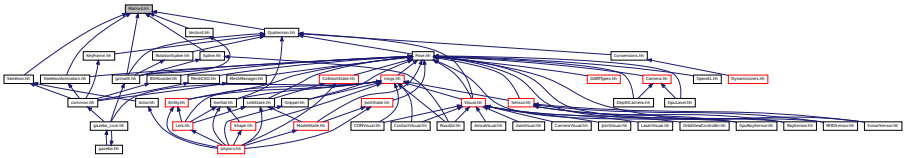
# 11.106 Matrix4.hh File Reference

```
#include <assert.h>
#include <iostream>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
```

Include dependency graph for Matrix4.hh:




This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::math::Matrix4**

  *A 3x3 matrix class.*

## Namespaces

- namespace **gazebo**

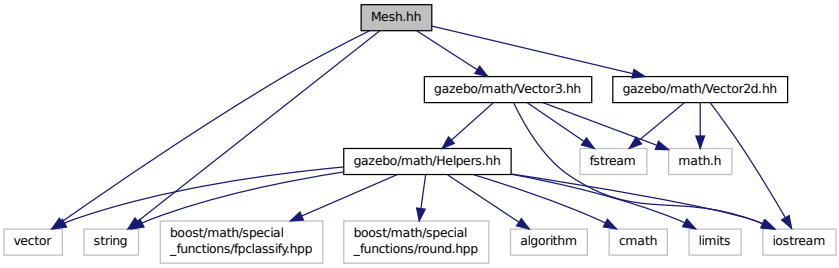
  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

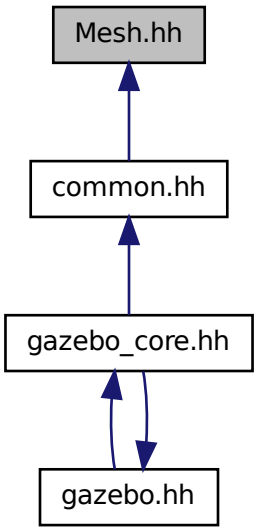
# 11.107 Mesh.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
```

Include dependency graph for Mesh.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::common::Mesh**

  *A 3D mesh.*

- struct **gazebo::common::NodeAssignment**

  *Vertex to node weighted assignement for skeleton animation visualization.*

- class **gazebo::common::SubMesh**

  *A child mesh.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

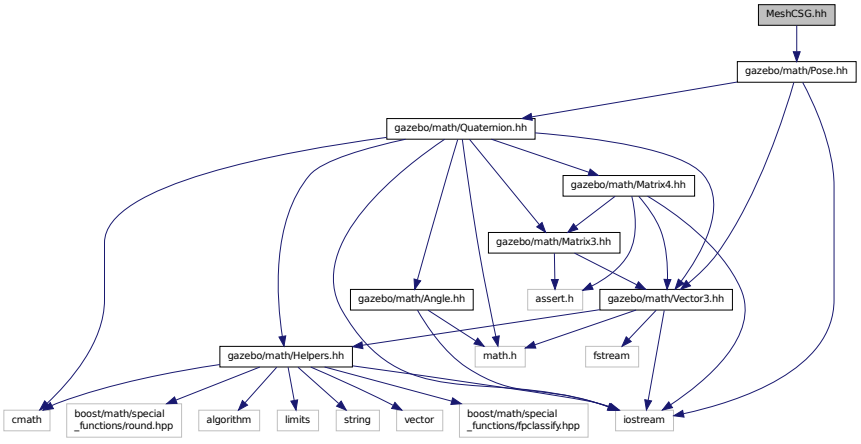
- namespace **gazebo::common**

  *Common namespace.*

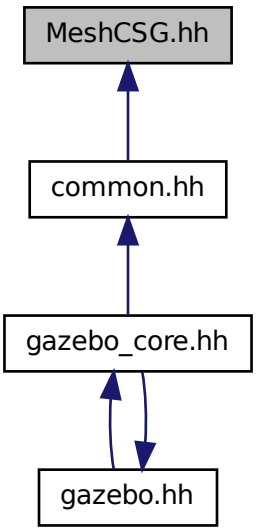
## 11.108 MeshCSG.hh File Reference

```
#include "gazebo/math/Pose.hh"
```

Include dependency graph for MeshCSG.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::common::MeshCSG**

  *Creates CSG meshes.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## Typedefs

- typedef _GPtrArray **GPtrArray**
- typedef _GtsSurface **GtsSurface**

### 11.108.1 Typedef Documentation

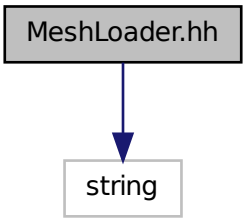
#### 11.108.1.1 typedef _GPtrArray GPtrArray

#### 11.108.1.2 typedef _GtsSurface GtsSurface

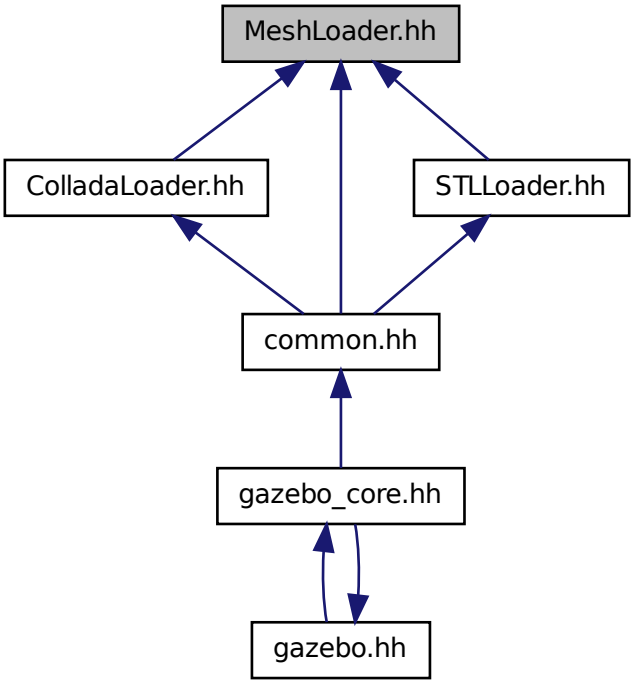
## 11.109 MeshLoader.hh File Reference

```
#include <string>
```

Include dependency graph for MeshLoader.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::MeshLoader**

  *Base class for loading meshes.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

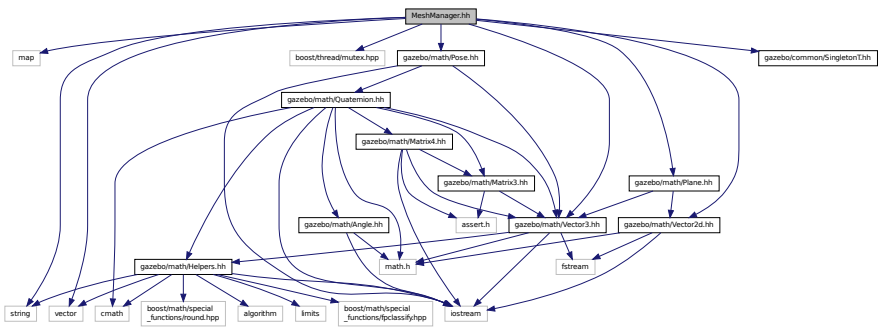
- namespace **gazebo::common**

  *Common namespace.*

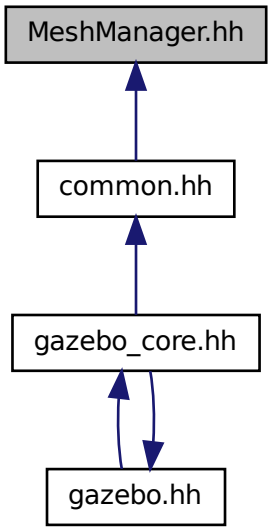
## 11.110 MeshManager.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include <boost/thread/mutex.hpp>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for MeshManager.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::MeshManager**

  *Maintains and manages all meshes.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

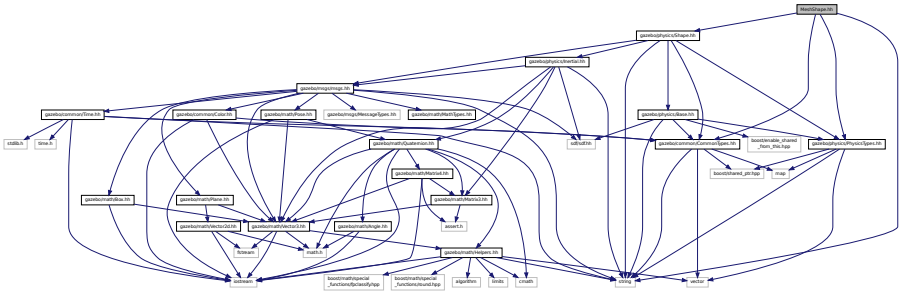
- namespace **gazebo::common**

  *Common namespace.*

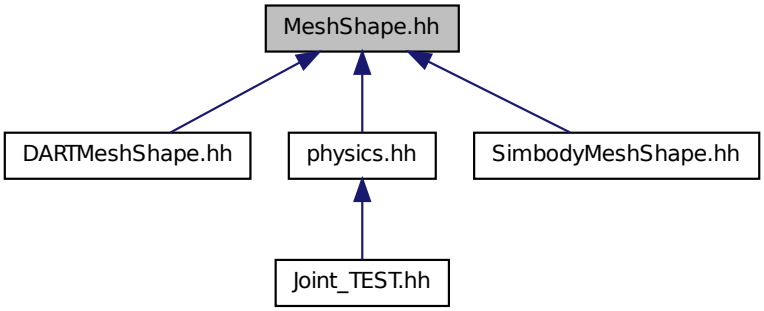
## 11.111 MeshShape.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for MeshShape.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::MeshShape**

  *Triangle mesh collision shape.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

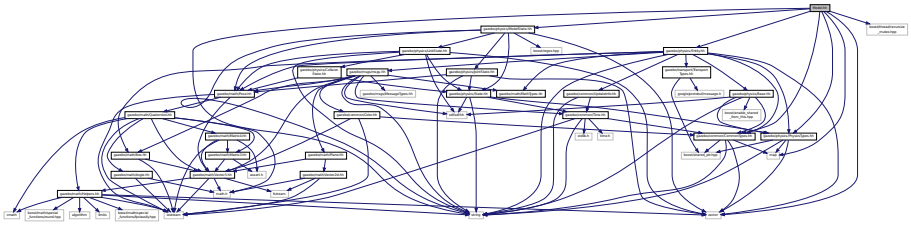
  *namespace for physics*

## 11.112 Model.hh File Reference

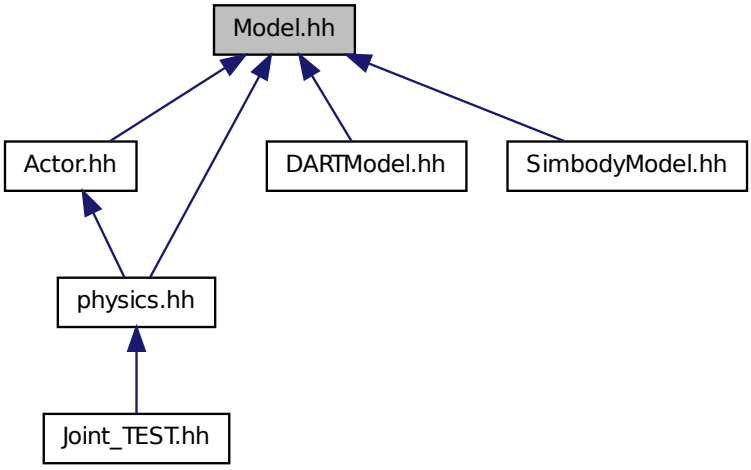
```
#include <string>
```

```
#include <map>
#include <vector>
#include <boost/thread/recursive_mutex.hpp>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/ModelState.hh"
#include "gazebo/physics/Entity.hh"
```

Include dependency graph for Model.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::physics::Model**

  *A model is a collection of links, joints, and plugins.*

## Namespaces

- namespace **boost**
- namespace **gazebo**

*Forward declarations for the common classes.*

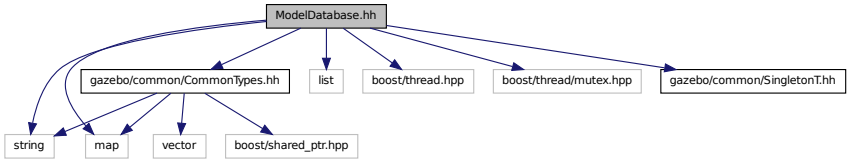
- namespace **gazebo::physics**

  *namespace for physics*

## 11.113 ModelDatabase.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include <boost/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for ModelDatabase.hh:




### Classes

- class **gazebo::common::ModelDatabase**

  *Connects to model database, and has utility functions to find models.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::common**

  *Common namespace.*

### Macros

- #define **GZ_MODEL_DB_MANIFEST_FILENAME** "database.config"

  *The file name of model database XML configuration.*

- #define **GZ_MODEL_MANIFEST_FILENAME** "model.config"

  *The file name of model XML configuration.*

### 11.113.1 Macro Definition Documentation

#### 11.113.1.1 #define GZ_MODEL_DB_MANIFEST_FILENAME "database.config"

The file name of model database XML configuration.

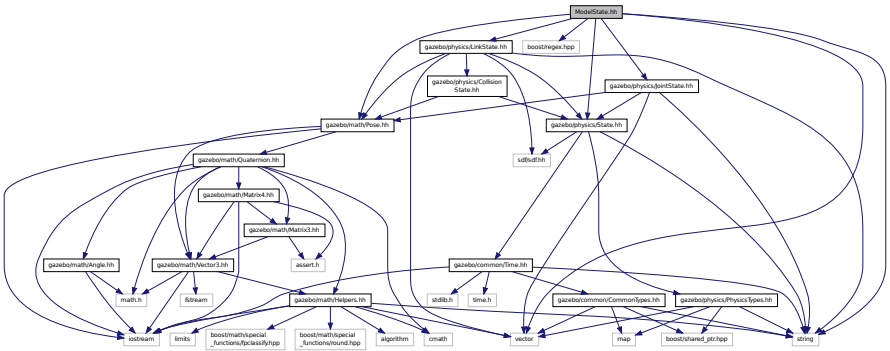
#### 11.113.1.2 #define GZ_MODEL_MANIFEST_FILENAME "model.config"

The file name of model XML configuration.

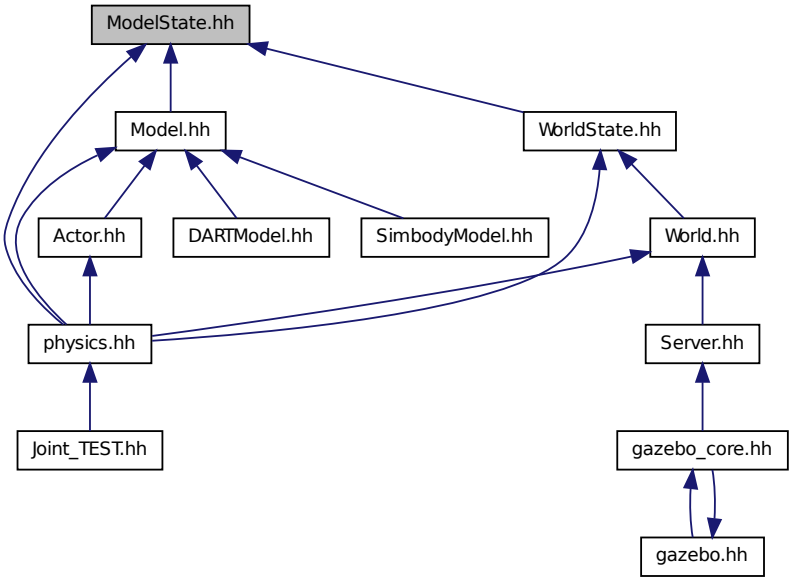
## 11.114 ModelState.hh File Reference

```
#include <vector>
#include <string>
#include <boost/regex.hpp>
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/State.hh"
#include "gazebo/physics/LinkState.hh"
#include "gazebo/physics/JointState.hh"
```

Include dependency graph for ModelState.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::ModelState**

  *Store state information of a **physics::Model** (p. 624) object.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

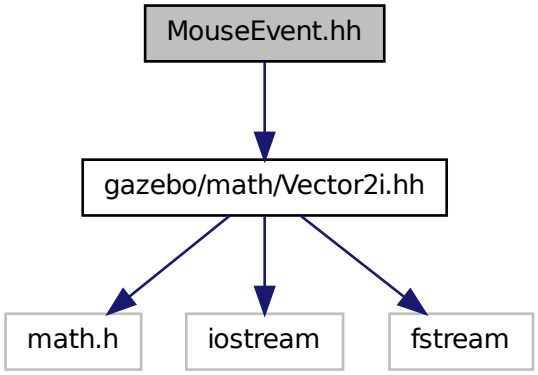
- namespace **gazebo::physics**

  *namespace for physics*

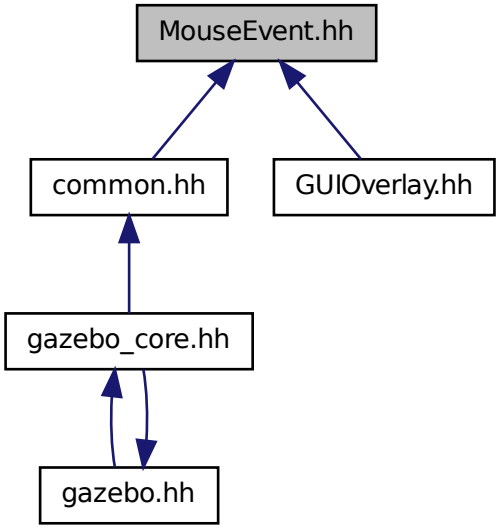
## 11.115 MouseEvent.hh File Reference

```
#include "gazebo/math/Vector2i.hh"
```

Include dependency graph for MouseEvent.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::common::MouseEvent**

  *Generic description of a mouse event.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::common**

  *Common namespace.*

## 11.116 MovableText.hh File Reference

```
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/math/MathTypes.hh"
```

Include dependency graph for MovableText.hh:

### Classes

- class **gazebo::rendering::MovableText**

  *Movable text.*

### Namespaces

- namespace **boost**
- namespace **gazebo**

  *Forward declarations for the common classes.*

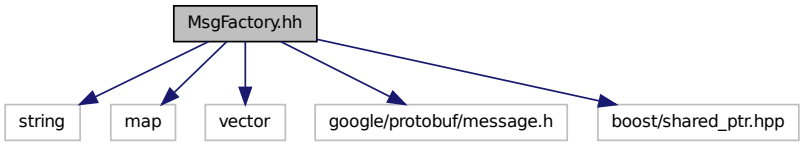
- namespace **gazebo::rendering**

  *Rendering namespace.*

## 11.117 MsgFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include <google/protobuf/message.h>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for MsgFactory.hh:




This graph shows which files directly or indirectly include this file:

### Classes

- class **gazebo::msgs::MsgFactory**

  *A factory that generates protobuf message based on a string type.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::msgs**

  *Messages namespace.*

### Macros

- #define **GZ_REGISTER_STATIC_MSG**(_msgtype, _classname)

  *Static message registration macro.*

### Typedefs

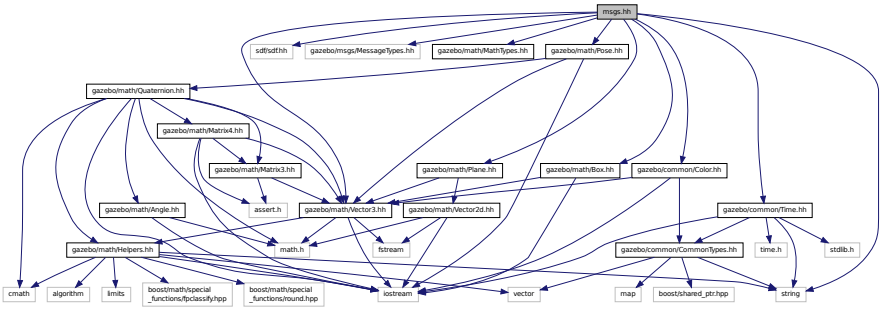
- typedef boost::shared_ptr
  < google::protobuf::Message >(∗ **gazebo::msgs::MsgFactoryFn** )()

## 11.118 msgs.hh File Reference

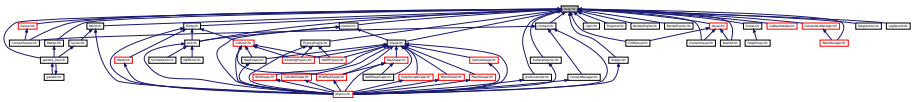
```
#include <string>
```

```
#include <sdf/sdf.hh>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/common/Time.hh"
```

Include dependency graph for msgs.hh:




This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::msgs**

  *Messages namespace.*

## Functions

- msgs::Vector3d **gazebo::msgs::Convert** (const math::Vector3 &_v)

  *Convert a **math::Vector3** (p. 1091) to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion** (p. 761) to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)

  *Convert a **math::Pose** (p. 734) to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)

  *Convert a **common::Color** (p. 233) to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

*Convert a **common::Time** (p. 1031) to a msgs::Time.*

- msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)

  *Convert a **math::Plane** (p. 726) to a msgs::PlaneGeom.*
- math::Vector3 **gazebo::msgs::Convert** (const msgs::Vector3d &_v)

  *Convert a msgs::Vector3d to a math::Vector.*
- math::Quaternion **gazebo::msgs::Convert** (const msgs::Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion** (p. 761).*
- math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 734).*
- common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 233).*
- common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 1031).*
- math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

  *Convert a msgs::PlaneGeom to a common::Plane.*
- msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")

  *Create a request message.*
- msgs::Fog **gazebo::msgs::FogFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Fog from a fog SDF element.*
- msgs::Geometry **gazebo::msgs::GeometryFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Geometry from a geometry SDF element.*
- msgs::Header * **gazebo::msgs::GetHeader** (google::protobuf::Message &_message)

  *Get the header from a protobuf message.*
- msgs::GUI **gazebo::msgs::GUIFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::GUI from a GUI SDF element.*
- void **gazebo::msgs::Init** (google::protobuf::Message &_message, const std::string &_id="")

  *Initialize a message.*
- msgs::Light **gazebo::msgs::LightFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Light from a light SDF element.*
- msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::MeshGeom from a mesh SDF element.*
- msgs::Scene **gazebo::msgs::SceneFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Scene from a scene SDF element.*
- void **gazebo::msgs::Set** (common::Image &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 474).*
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 474).*
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 1091).*
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 1091).*
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 761).*
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 734).*
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 233).*

- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 1031).*

- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 726).*

- void **gazebo::msgs::Stamp** (msgs::Header *_header)

  *Time stamp a header.*

- void **gazebo::msgs::Stamp** (msgs::Time *_time)

  *Set the time in a time message.*

- msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::TrackVisual from a track visual SDF element.*

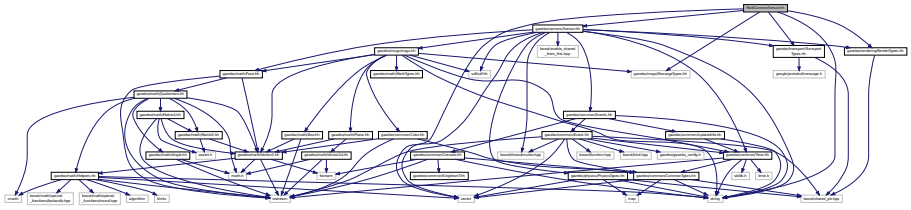
- msgs::Visual **gazebo::msgs::VisualFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::Visual from a visual SDF element.*

## 11.119 MultiCameraSensor.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for MultiCameraSensor.hh:



### Classes

- class **gazebo::sensors::MultiCameraSensor**

  *Multiple camera sensor.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

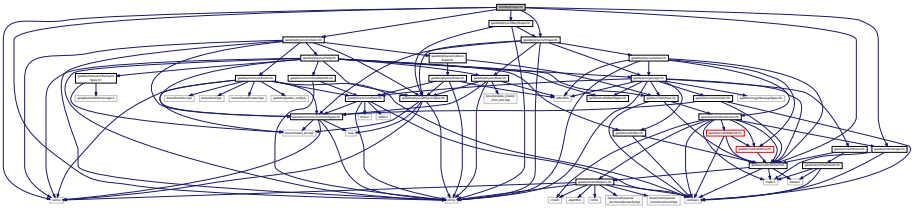
- namespace **gazebo::sensors**

  *Sensors namespace.*

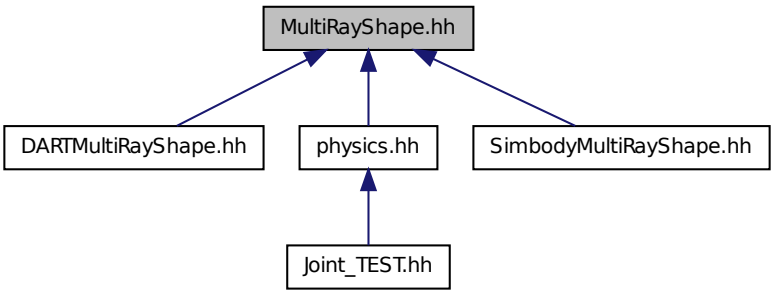
## 11.120 MultiRayShape.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/RayShape.hh"
```

Include dependency graph for MultiRayShape.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::MultiRayShape**

  *Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.*

### Namespaces

- namespace **gazebo**

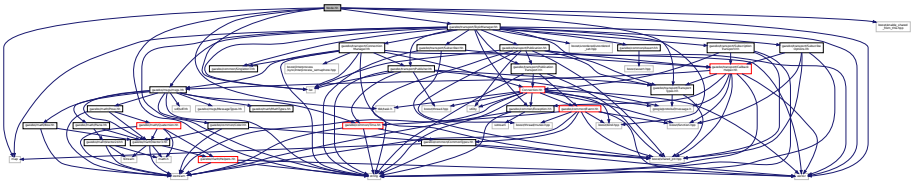
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

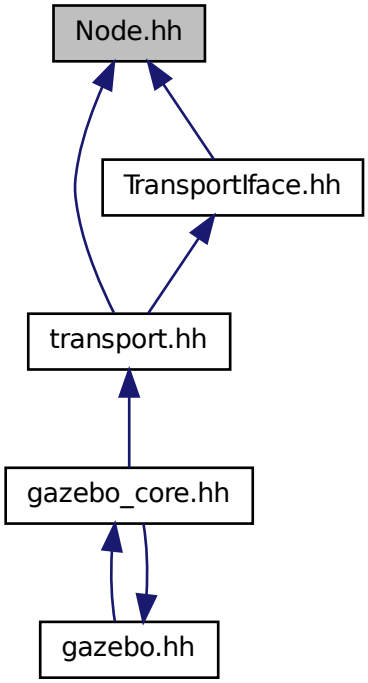
## 11.121 Node.hh File Reference

```
#include <tbb/task.h>
#include <boost/enable_shared_from_this.hpp>
#include <map>
#include <list>
#include <string>
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/TopicManager.hh"
```

Include dependency graph for Node.hh:



This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::transport::Node**

  *A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*

### Namespaces

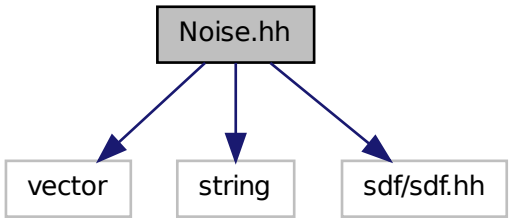
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

## 11.122 Noise.hh File Reference

```
#include <vector>
#include <string>
#include <sdf/sdf.hh>
```

Include dependency graph for Noise.hh:




### Classes

- class **gazebo::sensors::Noise**

  ***Noise*** (p. *689*) *models for sensor output signals.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

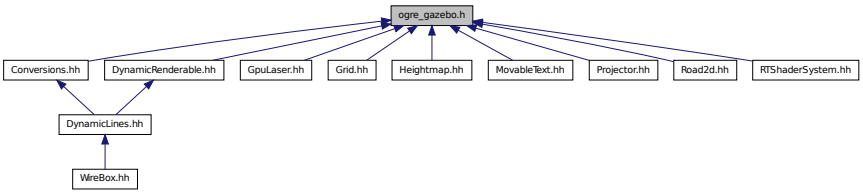
  *Sensors namespace.*

## 11.123 ogre_gazebo.h File Reference

```
#include <OGRE/Ogre.h>
#include <OGRE/OgreImageCodec.h>
#include <OGRE/OgreMovableObject.h>
#include <OGRE/OgreRenderable.h>
#include <OGRE/OgrePlugin.h>
#include <OGRE/OgreDataStream.h>
#include <OGRE/OgreLogManager.h>
#include <OGRE/OgreWindowEventUtilities.h>
#include <OGRE/OgreSceneQuery.h>
#include <OGRE/OgreRoot.h>
#include <OGRE/OgreSceneManager.h>
#include <OGRE/OgreSceneNode.h>
#include <OGRE/OgreVector3.h>
#include <OGRE/OgreManualObject.h>
#include <OGRE/OgreMaterialManager.h>
#include <OGRE/OgreColourValue.h>
#include <OGRE/OgreQuaternion.h>
#include <OGRE/OgreMesh.h>
#include <OGRE/OgreHardwareBufferManager.h>
#include <OGRE/OgreCamera.h>
#include <OGRE/OgreNode.h>
#include <OGRE/OgreSimpleRenderable.h>
#include <OGRE/OgreFrameListener.h>
#include <OGRE/OgreTexture.h>
#include <OGRE/OgreRenderObjectListener.h>
#include <OGRE/OgreTechnique.h>
#include <OGRE/OgrePass.h>
#include <OGRE/OgreTextureUnitState.h>
#include <OGRE/OgreGpuProgramManager.h>
#include <OGRE/OgreHighLevelGpuProgramManager.h>
#include <OGRE/OgreHardwarePixelBuffer.h>
#include <OGRE/OgreShadowCameraSetupPSSM.h>
#include <OGRE/Paging/OgrePageManager.h>
#include <OGRE/Paging/OgrePagedWorld.h>
#include <OGRE/Terrain/OgreTerrainPaging.h>
#include <OGRE/Terrain/OgreTerrainMaterialGeneratorA.h>
#include <OGRE/Terrain/OgreTerrain.h>
#include <OGRE/Terrain/OgreTerrainGroup.h>
#include <OGRE/OgreFontManager.h>
```

Include dependency graph for ogre_gazebo.h:

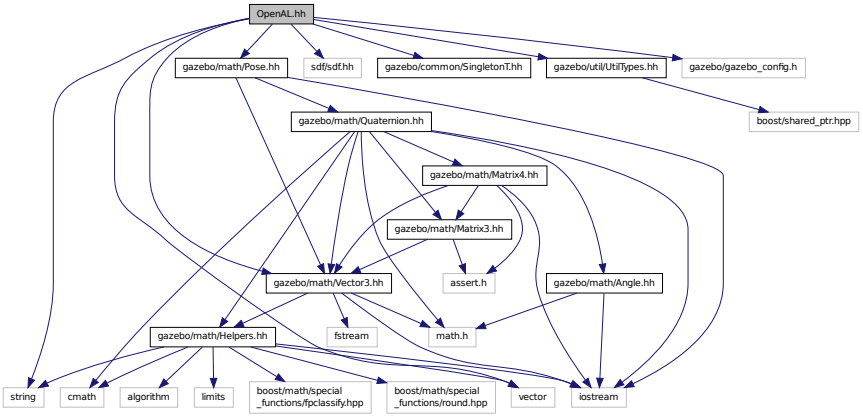
This graph shows which files directly or indirectly include this file:




## 11.124 OpenAL.hh File Reference

```
#include <string>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/util/UtilTypes.hh"
#include "gazebo/gazebo_config.h"
```

Include dependency graph for OpenAL.hh:




### Classes

- class **gazebo::util::OpenAL**

  *3D audio setup and playback.*

- class **gazebo::util::OpenALSink**

  ***OpenAL*** *(p. 695) Listener.*

- class **gazebo::util::OpenALSource**

  ***OpenAL*** *(p. 695) Source.*

### Namespaces

- namespace **gazebo**

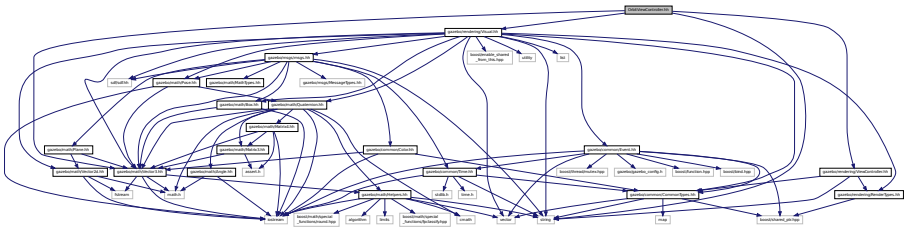
  *Forward declarations for the common classes.*

- namespace **gazebo::util**

## 11.125 OrbitViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/ViewController.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for OrbitViewController.hh:



### Classes

- class **gazebo::rendering::OrbitViewController**

  *Orbit view controller.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

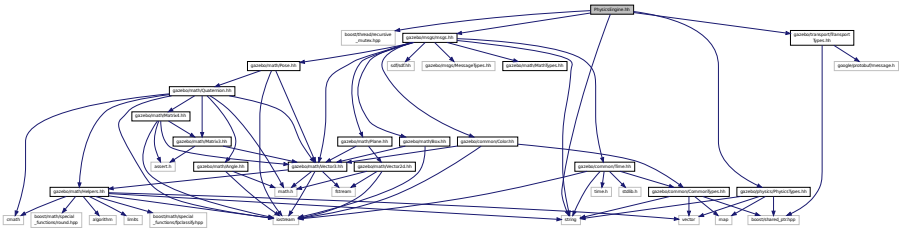
- namespace **gazebo::rendering**

  *Rendering namespace.*

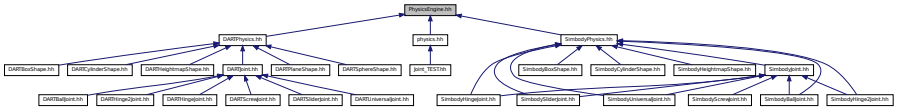
## 11.126 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::physics::PhysicsEngine**

  ***Base*** *(*p. *159) class for a physics engine.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

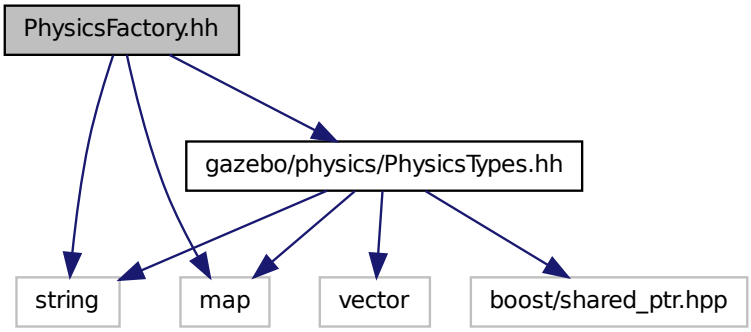
- namespace **gazebo::physics**

  *namespace for physics*

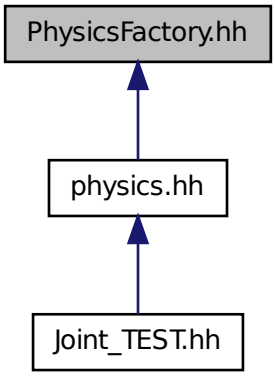
# 11.127 PhysicsFactory.hh File Reference

```
#include <string>
#include <map>
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsFactory.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::physics::PhysicsFactory**

  *The physics factory instantiates different physics engines.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::physics**

  *namespace for physics*

### Macros

- #define **GZ_REGISTER_PHYSICS_ENGINE**(name, classname)

  *Static physics registration macro.*

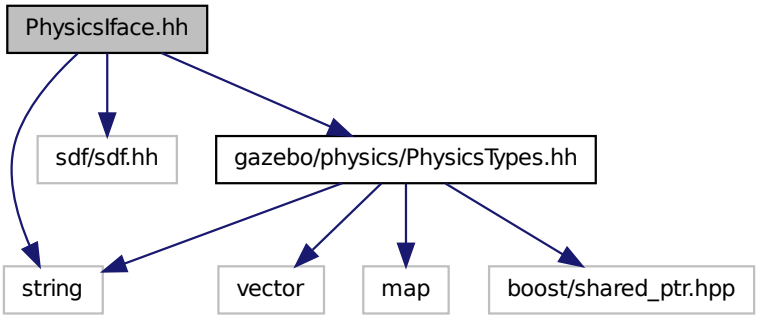
### Typedefs

- typedef PhysicsEnginePtr(∗ **gazebo::physics::PhysicsFactoryFn** )(WorldPtr world)

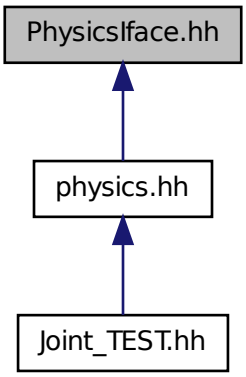
## 11.128 PhysicsIface.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsIface.hh:



This graph shows which files directly or indirectly include this file:




## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## Functions

- WorldPtr **gazebo::physics::create_world** (const std::string &_name="")

  *Create a world given a name.*
- bool **gazebo::physics::fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 94).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")

  *Returns a pointer to a world by name.*
- uint32_t **gazebo::physics::getUniqueId** ()

  *Get a unique ID.*
- void **gazebo::physics::init_world** (WorldPtr _world)

  *Init world given a pointer to it.*
- void **gazebo::physics::init_worlds** ()

  *initialize multiple worlds stored in static variable gazebo::g_worlds*
- bool **gazebo::physics::load** ()

  *Setup **gazebo::SystemPlugin** (p. 1030)'s and call **gazebo::transport::init** (p. 95).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)

  *Load world from sdf::Element pointer.*
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)

  *load multiple worlds from single sdf::Element pointer*

- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)

  *Pause world by calling **World::SetPaused*** (p. *1167*).
- void **gazebo::physics::pause_worlds** (bool pause)

  *pause multiple worlds stored in static variable gazebo::g_worlds*
- void **gazebo::physics::remove_worlds** ()

  *remove multiple worlds stored in static variable gazebo::g_worlds*
- void **gazebo::physics::run_world** (WorldPtr _world, unsigned int _iterations=0)

  *Run world by calling **World::Run()*** (p. *1167*) *given a pointer to it.*
- void **gazebo::physics::run_worlds** (unsigned int _iterations=0)

  *Run multiple worlds stored in static variable gazebo::g_worlds.*
- void **gazebo::physics::stop_world** (WorldPtr _world)

  *Stop world by calling **World::Stop()*** (p. *1168*) *given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()

  *stop multiple worlds stored in static variable gazebo::g_worlds*
- bool **gazebo::physics::worlds_running** ()

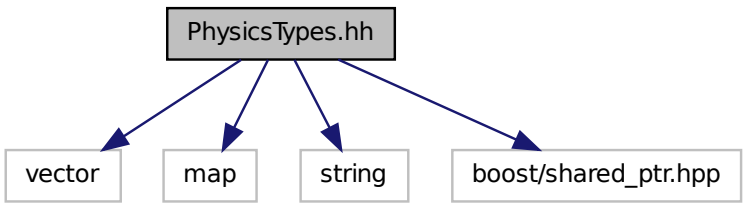
  *Return true if any world is running.*

## 11.129 PhysicsTypes.hh File Reference

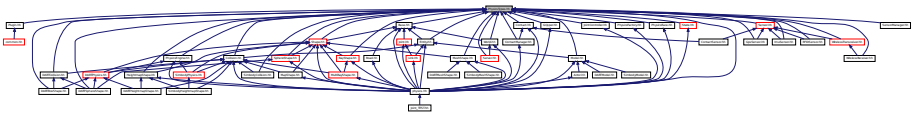
default namespace for gazebo

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for PhysicsTypes.hh:




This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## Macros

- #define **GZ_ALL_COLLIDE** 0x0FFFFFFF

  *Default collision bitmask.*
- #define **GZ_FIXED_COLLIDE** 0x00000001

  *Collision object will collide only with fixed objects.*
- #define **GZ_GHOST_COLLIDE** 0x10000000

  *Collides with everything else but other ghost.*
- #define **GZ_NONE_COLLIDE** 0x00000000

  *Collision object will collide with nothing.*
- #define **GZ_SENSOR_COLLIDE** 0x00000002

  *Collision object will collide only with sensors.*

## Typedefs

- typedef std::vector< ActorPtr > **gazebo::physics::Actor_V**
- typedef boost::shared_ptr< Actor > **gazebo::physics::ActorPtr**
- typedef std::vector< BasePtr > **gazebo::physics::Base_V**
- typedef boost::shared_ptr< Base > **gazebo::physics::BasePtr**
- typedef boost::shared_ptr < BoxShape > **gazebo::physics::BoxShapePtr**
- typedef std::vector< CollisionPtr > **gazebo::physics::Collision_V**
- typedef boost::shared_ptr < Collision > **gazebo::physics::CollisionPtr**
- typedef boost::shared_ptr < Contact > **gazebo::physics::ContactPtr**
- typedef boost::shared_ptr < CylinderShape > **gazebo::physics::CylinderShapePtr**
- typedef boost::shared_ptr< Entity > **gazebo::physics::EntityPtr**
- typedef boost::shared_ptr < Gripper > **gazebo::physics::GripperPtr**
- typedef boost::shared_ptr < HeightmapShape > **gazebo::physics::HeightmapShapePtr**
- typedef boost::shared_ptr < Inertial > **gazebo::physics::InertialPtr**
- typedef std::vector< JointPtr > **gazebo::physics::Joint_V**
- typedef std::vector < JointControllerPtr > **gazebo::physics::JointController_V**
- typedef boost::shared_ptr < JointController > **gazebo::physics::JointControllerPtr**
- typedef boost::shared_ptr< Joint > **gazebo::physics::JointPtr**
- typedef std::map< std::string, JointState > **gazebo::physics::JointState_M**

- typedef std::vector< LinkPtr > **gazebo::physics::Link_V**
- typedef boost::shared_ptr< Link > **gazebo::physics::LinkPtr**
- typedef std::map< std::string, LinkState > **gazebo::physics::LinkState_M**
- typedef boost::shared_ptr < MeshShape > **gazebo::physics::MeshShapePtr**
- typedef std::vector< ModelPtr > **gazebo::physics::Model_V**
- typedef boost::shared_ptr< Model > **gazebo::physics::ModelPtr**
- typedef std::map< std::string, ModelState > **gazebo::physics::ModelState_M**
- typedef boost::shared_ptr < MultiRayShape > **gazebo::physics::MultiRayShapePtr**
- typedef boost::shared_ptr < PhysicsEngine > **gazebo::physics::PhysicsEnginePtr**
- typedef boost::shared_ptr < RayShape > **gazebo::physics::RayShapePtr**
- typedef boost::shared_ptr< **Road** > **gazebo::physics::RoadPtr**
- typedef boost::shared_ptr< Shape > **gazebo::physics::ShapePtr**
- typedef boost::shared_ptr < SphereShape > **gazebo::physics::SphereShapePtr**
- typedef boost::shared_ptr < SurfaceParams > **gazebo::physics::SurfaceParamsPtr**
- typedef boost::shared_ptr< World > **gazebo::physics::WorldPtr**

### 11.129.1 Detailed Description

default namespace for gazebo

### 11.129.2 Macro Definition Documentation

#### 11.129.2.1 #define GZ_ALL_COLLIDE 0x0FFFFFFF

Default collision bitmask.

Collision objects will collide with everything.

#### 11.129.2.2 #define GZ_FIXED_COLLIDE 0x00000001

Collision object will collide only with fixed objects.

#### 11.129.2.3 #define GZ_GHOST_COLLIDE 0x10000000

Collides with everything else but other ghost.

#### 11.129.2.4 #define GZ_NONE_COLLIDE 0x00000000

Collision object will collide with nothing.

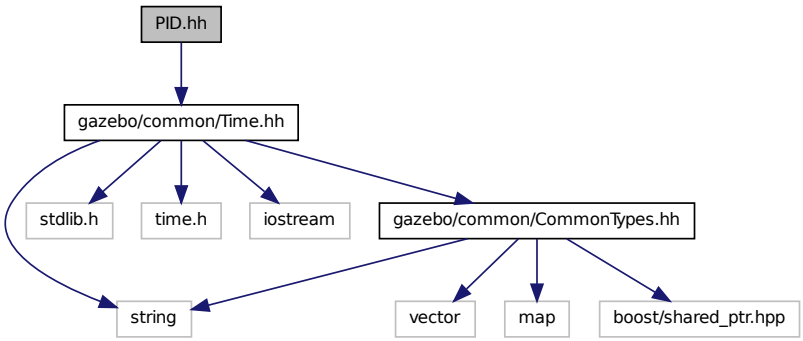
#### 11.129.2.5 #define GZ_SENSOR_COLLIDE 0x00000002

Collision object will collide only with sensors.

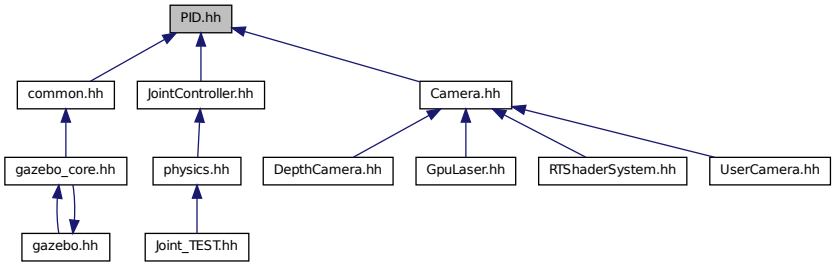
## 11.130 PID.hh File Reference

```
#include "gazebo/common/Time.hh"
```

Include dependency graph for PID.hh:




This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::PID**

  *Generic **PID** (p. 722) controller class.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

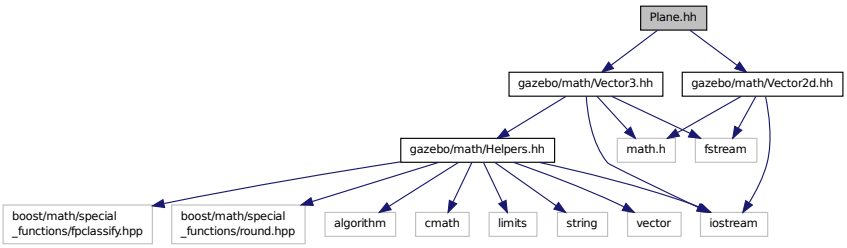
- namespace **gazebo::common**

  *Common namespace.*

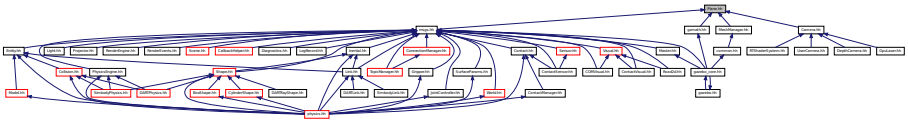
## 11.131 Plane.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
```

Include dependency graph for Plane.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::math::Plane**

  *A plane and related functions.*

### Namespaces

- namespace **gazebo**

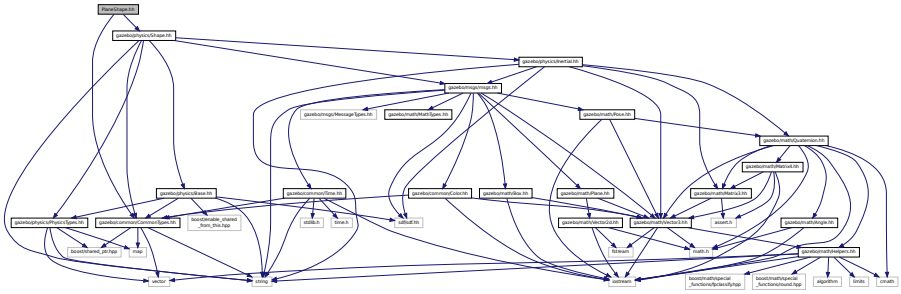
  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

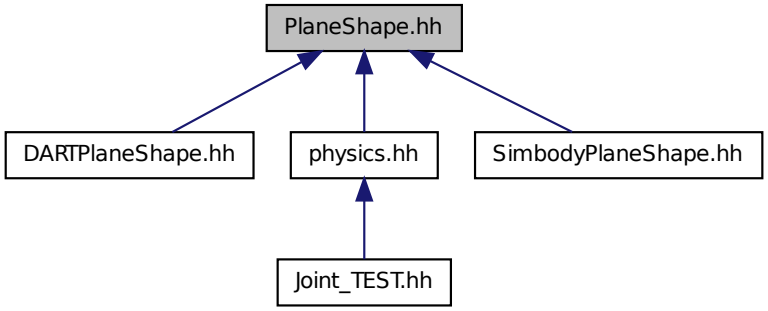
## 11.132 PlaneShape.hh File Reference

```
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for PlaneShape.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::physics::PlaneShape**

  ***Collision*** *(*p. *220) for an infinite plane.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

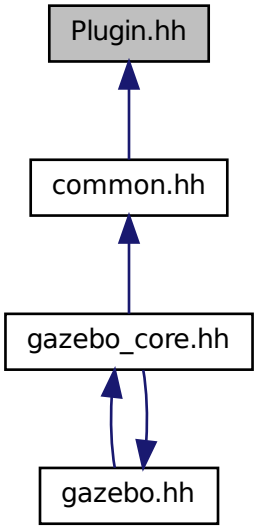
# 11.133 Plugin.hh File Reference

```
#include <unistd.h>
```

```
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo/gazebo_config.h>
#include <dlfcn.h>
#include <list>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/SystemPaths.hh"
#include "gazebo/common/Console.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for Plugin.hh:

This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::ModelPlugin**

*A plugin with access to **physics::Model** (p. 624).*

- class **gazebo::PluginT< T >**

  *A class which all plugins must inherit from.*

- class **gazebo::SensorPlugin**

  *A plugin with access to physics::Sensor.*

- class **gazebo::SystemPlugin**

  *A plugin loaded within the gzserver on startup.*

- class **gazebo::VisualPlugin**

  *A plugin loaded within the gzserver on startup.*

- class **gazebo::WorldPlugin**

  *A plugin with access to **physics::World** (p. 1157).*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

## Macros

- #define **GZ_REGISTER_MODEL_PLUGIN**(classname)

  *Plugin registration function for model plugin.*

- #define **GZ_REGISTER_SENSOR_PLUGIN**(classname)

  *Plugin registration function for sensors.*

- #define **GZ_REGISTER_SYSTEM_PLUGIN**(classname)

  *Plugin registration function for system plugin.*

- #define **GZ_REGISTER_VISUAL_PLUGIN**(classname)

  *Plugin registration function for visual plugin.*

- #define **GZ_REGISTER_WORLD_PLUGIN**(classname)

  *Plugin registration function for world plugin.*

## Enumerations

- enum **gazebo::PluginType** {
  **gazebo::WORLD_PLUGIN**, **gazebo::MODEL_PLUGIN**, **gazebo::SENSOR_PLUGIN**, **gazebo::SYSTEM_PLUGIN**,
  **gazebo::VISUAL_PLUGIN** }

  *Used to specify the type of plugin.*

### 11.133.1 Macro Definition Documentation

#### 11.133.1.1 #define GZ_REGISTER_MODEL_PLUGIN( *classname* )

**Value:**

```
extern "C" gazebo::ModelPlugin *RegisterPlugin();    gazebo::ModelPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for model plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

**Returns**

the name of the registered plugin

#### 11.133.1.2 #define GZ_REGISTER_SENSOR_PLUGIN( *classname* )

**Value:**

```
extern "C" gazebo::SensorPlugin *RegisterPlugin();    gazebo::SensorPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for sensors.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

**Returns**

the name of the registered plugin

#### 11.133.1.3 #define GZ_REGISTER_SYSTEM_PLUGIN( *classname* )

**Value:**

```
extern "C" gazebo::SystemPlugin *RegisterPlugin();    gazebo::SystemPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for system plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

**Returns**

the name of the registered plugin

#### 11.133.1.4 #define GZ_REGISTER_VISUAL_PLUGIN( *classname* )

**Value:**

```
extern "C" gazebo::VisualPlugin *RegisterPlugin();    gazebo::VisualPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for visual plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

**Returns**

the name of the registered plugin

#### 11.133.1.5 #define GZ_REGISTER_WORLD_PLUGIN( *classname* )

**Value:**

```
extern "C" gazebo::WorldPlugin *RegisterPlugin();    gazebo::WorldPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for world plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

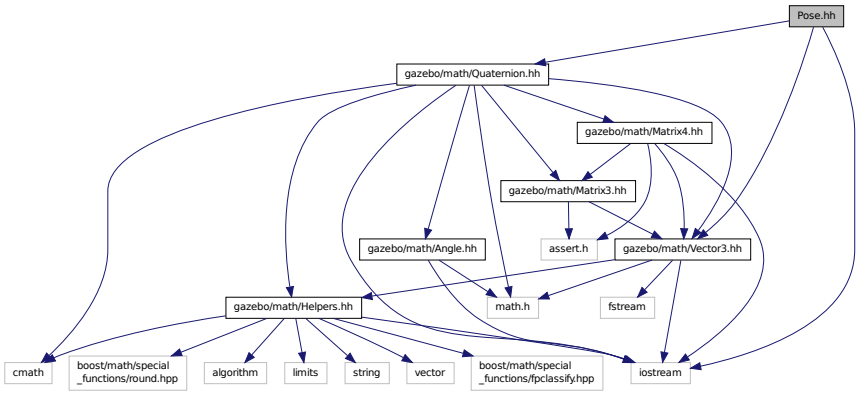
**Returns**

the name of the registered plugin

## 11.134 Pose.hh File Reference

```
#include <iostream>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh"
```

Include dependency graph for Pose.hh:




This graph shows which files directly or indirectly include this file:

### Classes

- class **gazebo::math::Pose**

  *Encapsulates a position and rotation in three space.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

## 11.135 Projector.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include <sdf/sdf.hh>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/transport.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for Projector.hh:

### Classes

- class **gazebo::rendering::Projector**

  *Projects a material onto surface, light a light projector.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

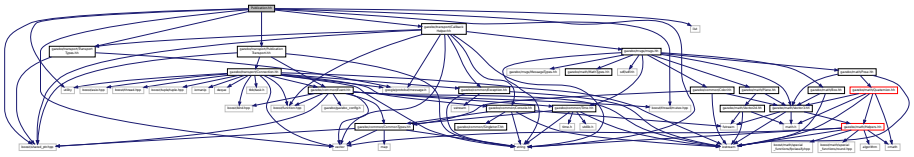
  *Rendering namespace.*

## 11.136 Publication.hh File Reference

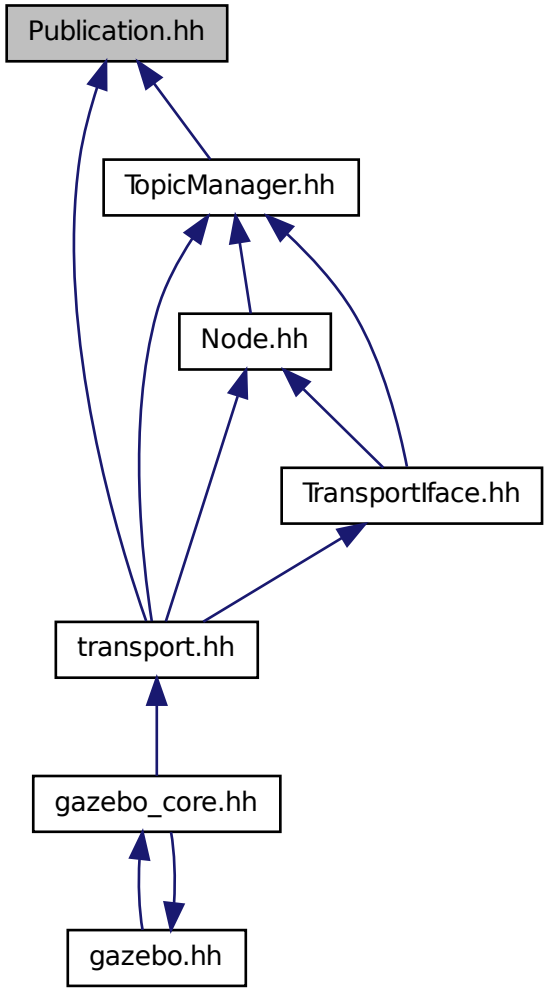
```
#include <utility>
```

```
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <list>
#include <string>
#include <vector>
#include "gazebo/transport/CallbackHelper.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/PublicationTransport.hh"
```

Include dependency graph for Publication.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::Publication**

  *A publication for a topic.*

## Namespaces

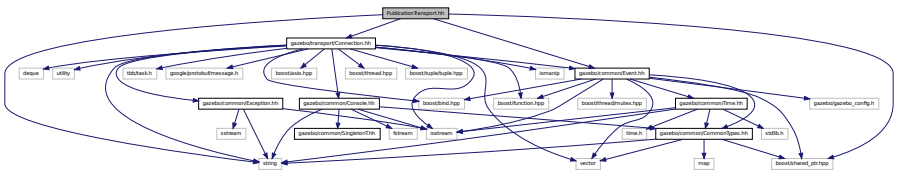
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

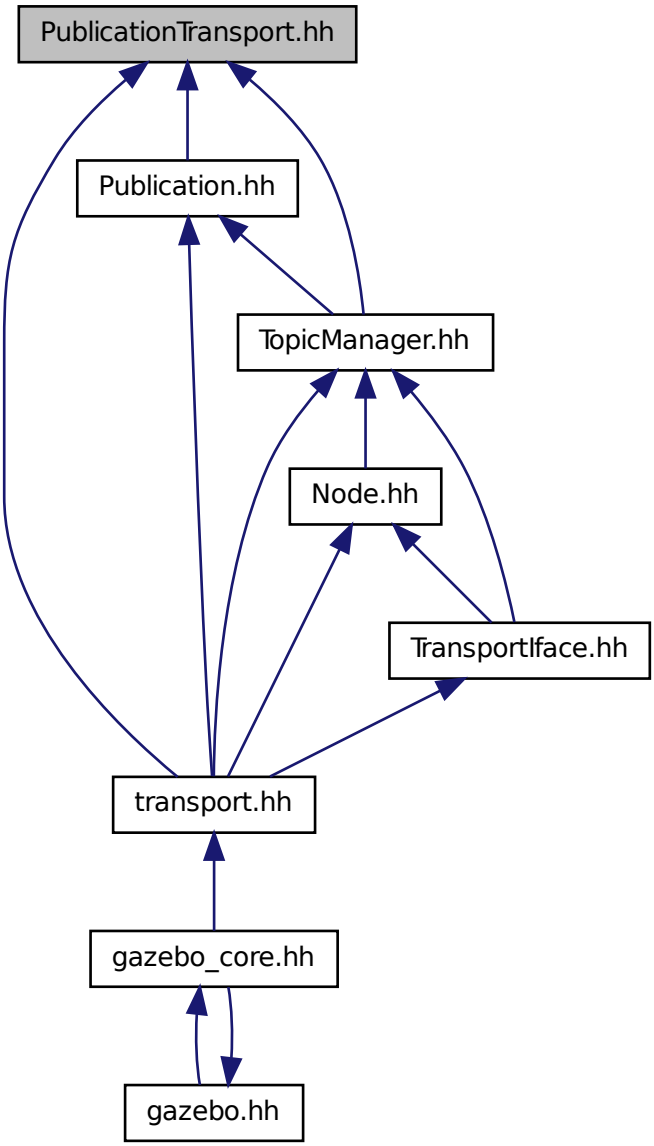
## 11.137 PublicationTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "gazebo/transport/Connection.hh"
#include "gazebo/common/Event.hh"
```

Include dependency graph for PublicationTransport.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::PublicationTransport**

  *transport/transport.hh*

### Namespaces

- namespace **gazebo**

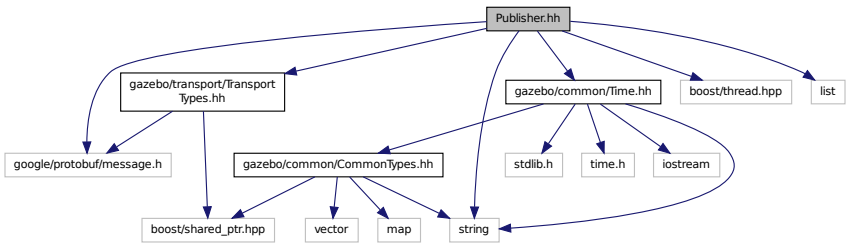
  *Forward declarations for the common classes.*

- namespace **gazebo::transport**

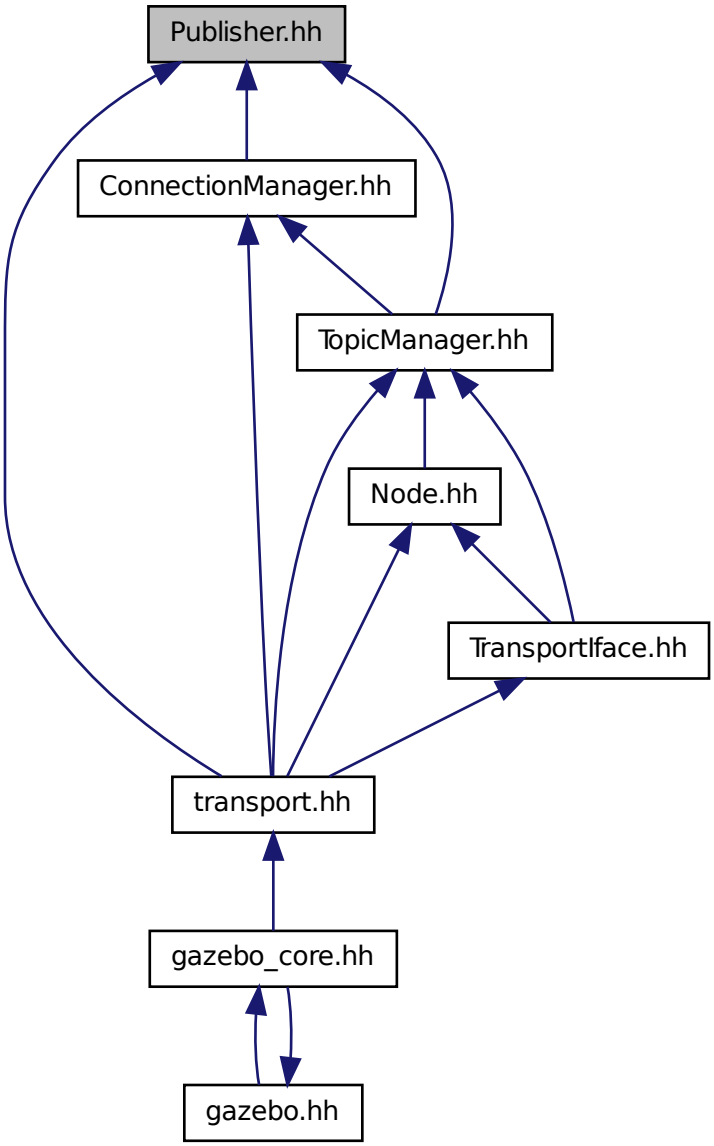
## 11.138 Publisher.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/thread.hpp>
#include <string>
#include <list>
#include "gazebo/common/Time.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for Publisher.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::Publisher**

  *A publisher of messages on a topic.*

### Namespaces

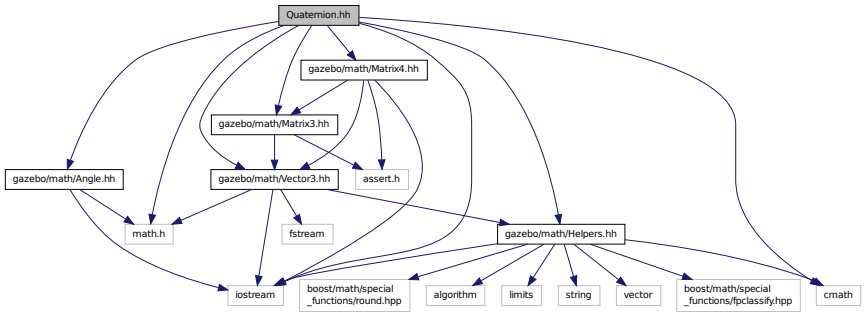
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

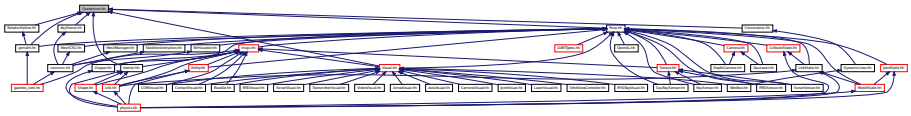
## 11.139 Quaternion.hh File Reference

```
#include <math.h>
#include <iostream>
#include <cmath>
#include "gazebo/math/Helpers.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
#include "gazebo/math/Matrix4.hh"
```

Include dependency graph for Quaternion.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::math::Quaternion**

  *A quaternion class.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

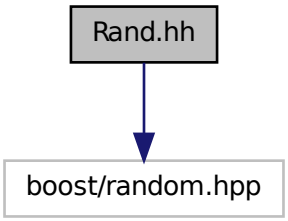
- namespace **gazebo::math**

  *Math namespace.*

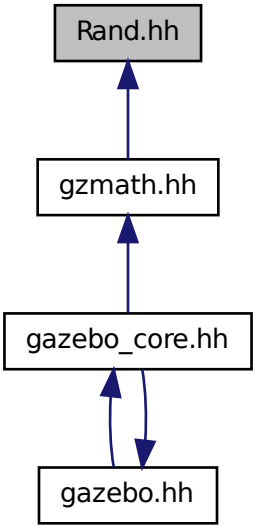
## 11.140 Rand.hh File Reference

```
#include <boost/random.hpp>
```

Include dependency graph for Rand.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::math::Rand**

  *Random number generator class.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

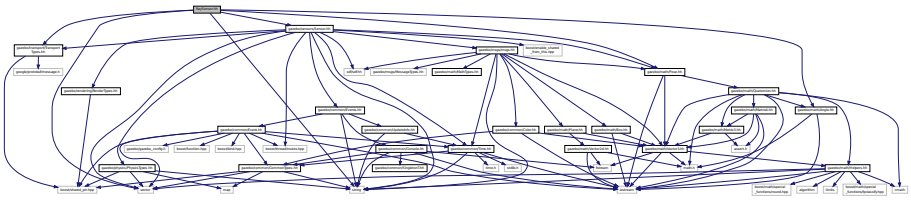
### Typedefs

- typedef boost::mt19937 **gazebo::math::GeneratorType**
- typedef boost::normal_distribution < double > **gazebo::math::NormalRealDist**
- typedef boost::variate_generator < GeneratorType &, NormalRealDist > **gazebo::math::NRealGen**
- typedef boost::variate_generator < GeneratorType &, UniformIntDist > **gazebo::math::UIntGen**
- typedef boost::uniform_int< int > **gazebo::math::UniformIntDist**
- typedef boost::uniform_real < double > **gazebo::math::UniformRealDist**
- typedef boost::variate_generator < GeneratorType &, UniformRealDist > **gazebo::math::URealGen**

## 11.141 RaySensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for RaySensor.hh:



### Classes

- class **gazebo::sensors::RaySensor**

  ***Sensor*** (p. *837) with one or more rays.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

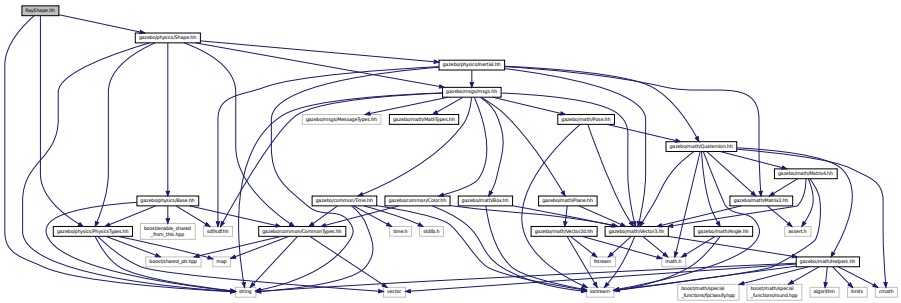
- namespace **gazebo::sensors**

  *Sensors namespace.*

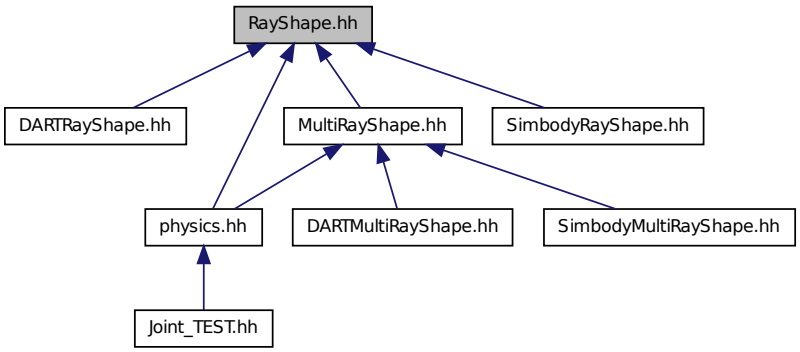
## 11.142 RayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for RayShape.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::RayShape**

  ***Base*** *(p. 159) class for Ray collision geometry.*

### Namespaces

- namespace **gazebo**

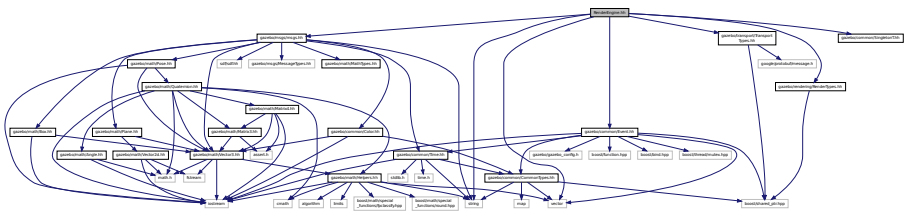
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.143 RenderEngine.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for RenderEngine.hh:



### Classes

- class **gazebo::rendering::RenderEngine**

  *Adaptor to Ogre3d.*

### Namespaces

- namespace **gazebo**

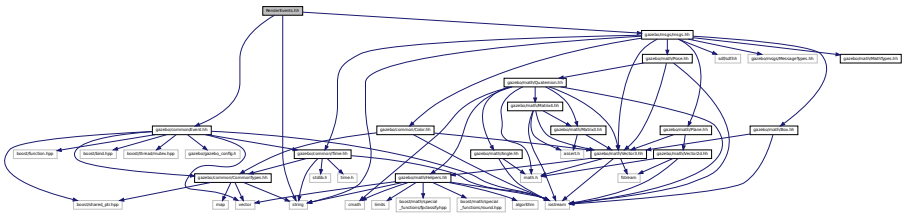
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*
- namespace **Ogre**

## 11.144 RenderEvents.hh File Reference

```
#include <string>
#include "gazebo/common/Event.hh"
#include "gazebo/msgs/msgs.hh"
```

Include dependency graph for RenderEvents.hh:



### Classes

- class **gazebo::rendering::Events**

  *Base class for rendering events.*

### Namespaces

- namespace **gazebo**

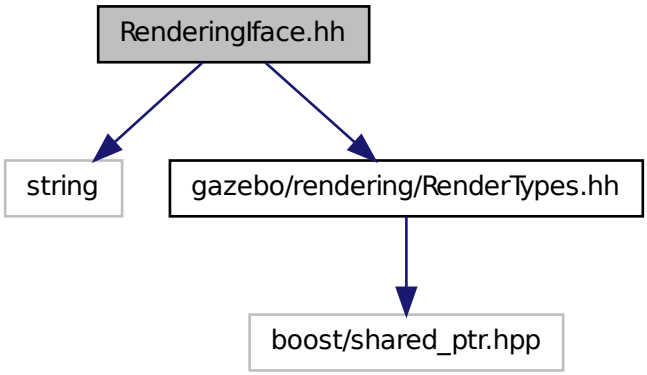
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

## 11.145 RenderingIface.hh File Reference

```
#include <string>
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for RenderingIface.hh:




## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::rendering**

  *Rendering namespace.*

## Functions

- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)

  *create **rendering::Scene*** (p. *814*) *by name.*

- bool **gazebo::rendering::fini** ()

  *teardown rendering engine.*

- rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")

  *get pointer to **rendering::Scene*** (p. *814*) *by name.*

- bool **gazebo::rendering::init** ()

  *init rendering engine.*

- bool **gazebo::rendering::load** ()

  *load rendering engine.*

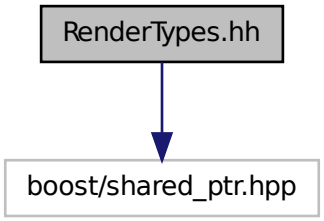
- void **gazebo::rendering::remove_scene** (const std::string &_name)

  *remove a **rendering::Scene*** (p. *814*) *by name*

## 11.146 RenderTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for RenderTypes.hh:




This graph shows which files directly or indirectly include this file:

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

### Macros

- #define **GZ_VISIBILITY_ALL** 0x0FFFFFFF

  *Render everything visibility mask.*
- #define **GZ_VISIBILITY_GUI** 0x00000001

  *Render GUI visuals mask.*
- #define **GZ_VISIBILITY_SELECTABLE** 0x00000002

  *Render visuals that are selectable mask.*
- #define **GZ_VISIBILITY_SELECTION** 0x10000000

  *Renders only objects that can be selected.*

### Typedefs

- typedef boost::shared_ptr
  < ArrowVisual > **gazebo::rendering::ArrowVisualPtr**

- typedef boost::shared_ptr < AxisVisual > **gazebo::rendering::AxisVisualPtr**
- typedef boost::shared_ptr< Camera > **gazebo::rendering::CameraPtr**
- typedef boost::shared_ptr < CameraVisual > **gazebo::rendering::CameraVisualPtr**
- typedef boost::shared_ptr < COMVisual > **gazebo::rendering::COMVisualPtr**
- typedef boost::shared_ptr < ContactVisual > **gazebo::rendering::ContactVisualPtr**
- typedef boost::shared_ptr < DepthCamera > **gazebo::rendering::DepthCameraPtr**
- typedef boost::shared_ptr < DynamicLines > **gazebo::rendering::DynamicLinesPtr**
- typedef boost::shared_ptr < GpuLaser > **gazebo::rendering::GpuLaserPtr**
- typedef boost::shared_ptr < JointVisual > **gazebo::rendering::JointVisualPtr**
- typedef boost::shared_ptr < LaserVisual > **gazebo::rendering::LaserVisualPtr**
- typedef boost::shared_ptr< Light > **gazebo::rendering::LightPtr**
- typedef boost::shared_ptr < RFIDTagVisual > **gazebo::rendering::RFIDTagVisualPtr**
- typedef boost::shared_ptr < RFIDVisual > **gazebo::rendering::RFIDVisualPtr**
- typedef boost::shared_ptr< Scene > **gazebo::rendering::ScenePtr**
- typedef boost::shared_ptr < SelectionObj > **gazebo::rendering::SelectionObjPtr**
- typedef boost::shared_ptr < SonarVisual > **gazebo::rendering::SonarVisualPtr**
- typedef boost::shared_ptr < UserCamera > **gazebo::rendering::UserCameraPtr**
- typedef boost::shared_ptr< Visual > **gazebo::rendering::VisualPtr**
- typedef boost::shared_ptr < WindowManager > **gazebo::rendering::WindowManagerPtr**
- typedef boost::shared_ptr < WrenchVisual > **gazebo::rendering::WrenchVisualPtr**

## Enumerations

- enum **gazebo::rendering::RenderOpType** { **gazebo::rendering::RENDERING_POINT_LIST** = 0, **gazebo::rendering::RENDERING_LINE_LIST** = 1, **gazebo::rendering::RENDERING_LINE_STRIP** = 2, **gazebo::rendering::RENDERING_TRIANGLE_LIST** = 3, **gazebo::rendering::RENDERING_TRIANGLE_STRIP** = 4, **gazebo::rendering::RENDERING_TRIANGLE_FAN** = 5, **gazebo::rendering::RENDERING_MESH_RESOURCE** = 6 }

  *Type of render operation for a drawable.*

### 11.146.1 Macro Definition Documentation

#### 11.146.1.1 #define GZ_VISIBILITY_ALL 0x0FFFFFFF

Render everything visibility mask.

#### 11.146.1.2 #define GZ_VISIBILITY_GUI 0x00000001

Render GUI visuals mask.

#### 11.146.1.3 #define GZ_VISIBILITY_SELECTABLE 0x00000002

Render visuals that are selectable mask.

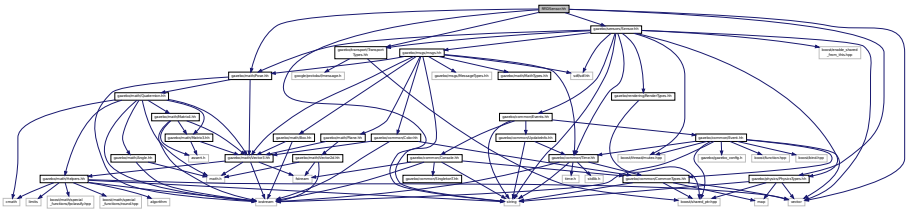
#### 11.146.1.4 #define GZ_VISIBILITY_SELECTION 0x10000000

Renders only objects that can be selected.

## 11.147 RFIDSensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for RFIDSensor.hh:



### Classes

- class **gazebo::sensors::RFIDSensor**

  ***Sensor*** *(p. 837) class for RFID type of sensor.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

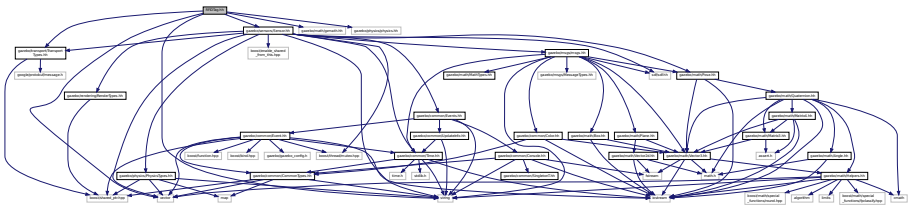
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.148 RFIDTag.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/math/gzmath.hh"
#include "gazebo/physics/physics.hh"
```

Include dependency graph for RFIDTag.hh:



### Classes

- class **gazebo::sensors::RFIDTag**

  ***RFIDTag*** (p. *798) to interact with RFIDTagSensors.*

### Namespaces

- namespace **gazebo**

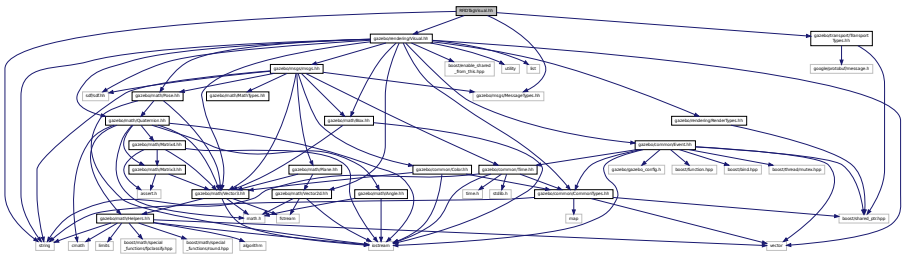
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.149 RFIDTagVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for RFIDTagVisual.hh:



### Classes

- class **gazebo::rendering::RFIDTagVisual**

  *Visualization for RFID tags sensor.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

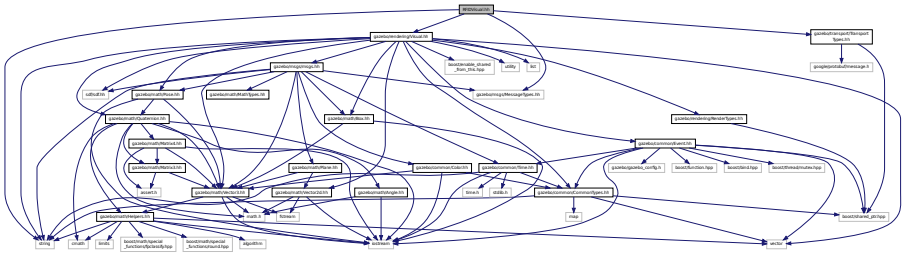
- namespace **gazebo::rendering**

  *Rendering namespace.*

## 11.150 RFIDVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for RFIDVisual.hh:



### Classes

- class **gazebo::rendering::RFIDVisual**

  *Visualization for RFID sensor.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

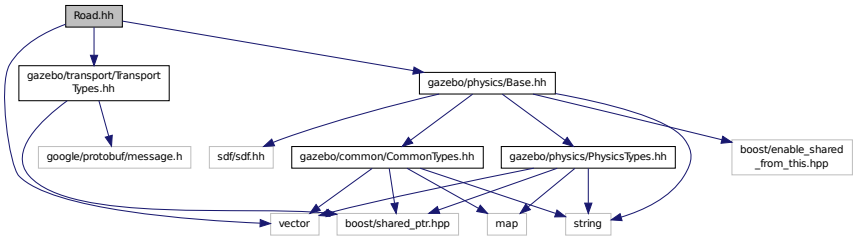
- namespace **gazebo::rendering**

  *Rendering namespace.*

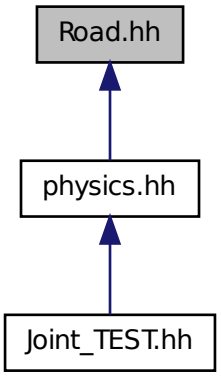
## 11.151 Road.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Road.hh:




This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::Road**
  *for building a **Road** (p. 804) from SDF*

### Namespaces

- namespace **gazebo**
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**
  *namespace for physics*

## 11.152 Road2d.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Spline.hh"
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for Road2d.hh:

### Classes

- class **gazebo::rendering::Road2d**

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

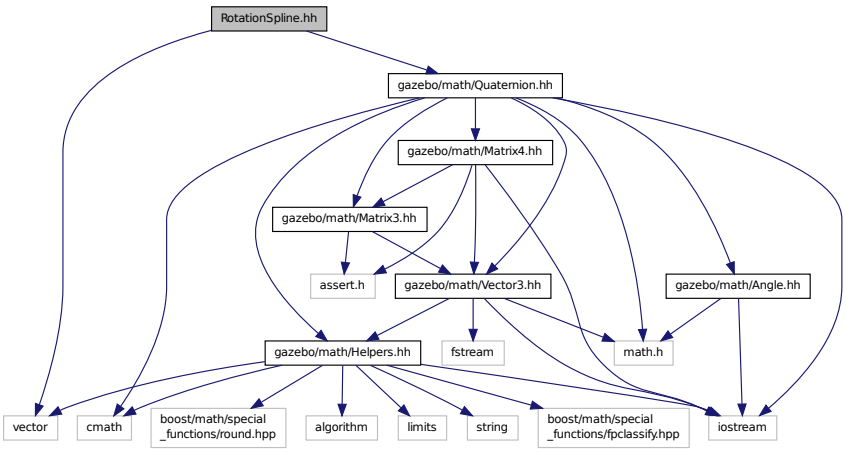
- namespace **gazebo::rendering**

  *Rendering namespace.*

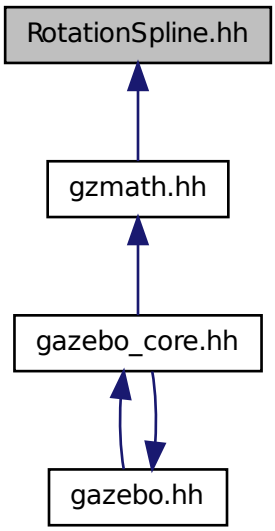
## 11.153 RotationSpline.hh File Reference

```
#include <vector>
#include "gazebo/math/Quaternion.hh"
```

Include dependency graph for RotationSpline.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::math::RotationSpline**

  ***Spline*** *(p. 993) for rotations.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::math**

  *Math namespace.*

## 11.154 RTShaderSystem.hh File Reference

```
#include <list>
#include <string>
#include <vector>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/gazebo_config.h"
#include "gazebo/rendering/Camera.hh"
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for RTShaderSystem.hh:

### Classes

- class **gazebo::rendering::RTShaderSystem**

  *Implements **Ogre** (p. 130)'s Run-Time Shader system.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::rendering**

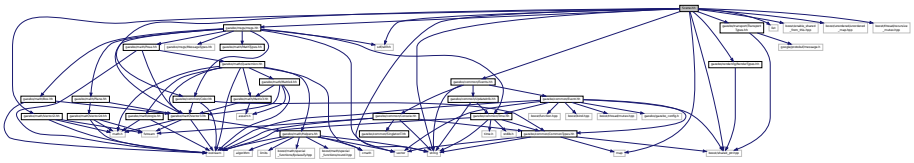
  *Rendering namespace.*

## 11.155 Scene.hh File Reference

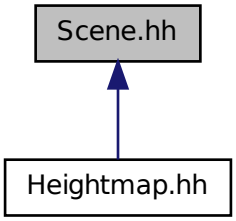
```
#include <vector>
```

```
#include <map>
#include <string>
#include <list>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/unordered/unordered_map.hpp>
#include <boost/thread/recursive_mutex.hpp>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/math/Vector2i.hh"
```

Include dependency graph for Scene.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::rendering::Scene**

  *Representation of an entire scene graph.*

## Namespaces

- namespace **boost**
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

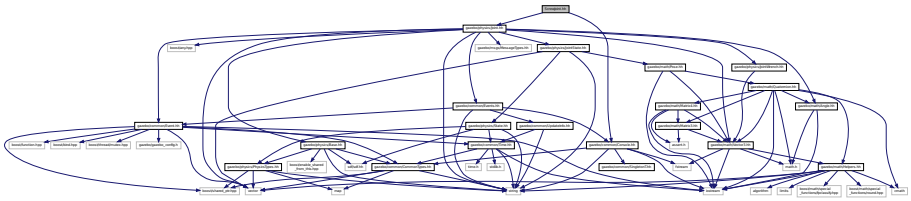
*Rendering namespace.*

- namespace **Ogre**
- namespace **SkyX**

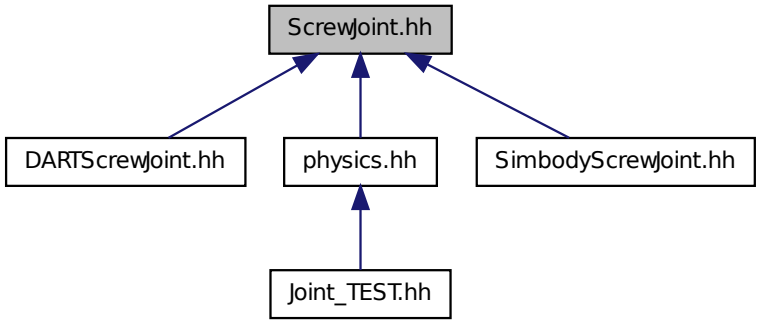
## 11.156 ScrewJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
#include "gazebo/common/Console.hh"
```

Include dependency graph for ScrewJoint.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::ScrewJoint**< **T** >

  *A screw joint, which has both prismatic and rotational DOFs.*

### Namespaces

- namespace **gazebo**

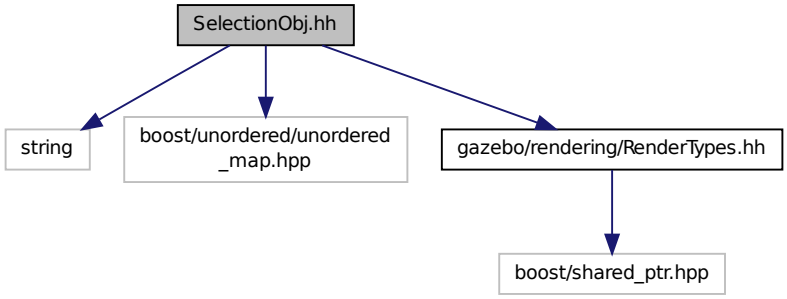
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.157 SelectionObj.hh File Reference

```
#include <string>
#include <boost/unordered/unordered_map.hpp>
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for SelectionObj.hh:




### Classes

- class **gazebo::rendering::SelectionObj**

  *Interactive selection object for models and links.*

### Namespaces

- namespace **gazebo**

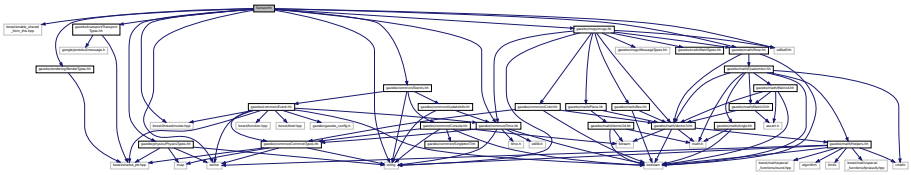
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

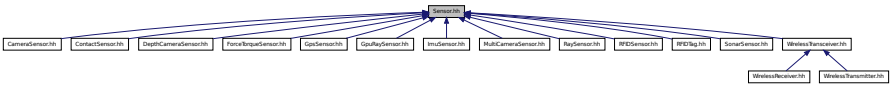
## 11.158 Sensor.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <boost/thread/mutex.hpp>
#include <vector>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for Sensor.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::sensors::Sensor**

  *Base class for sensors.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::sensors**

  *Sensors namespace.*

## Enumerations

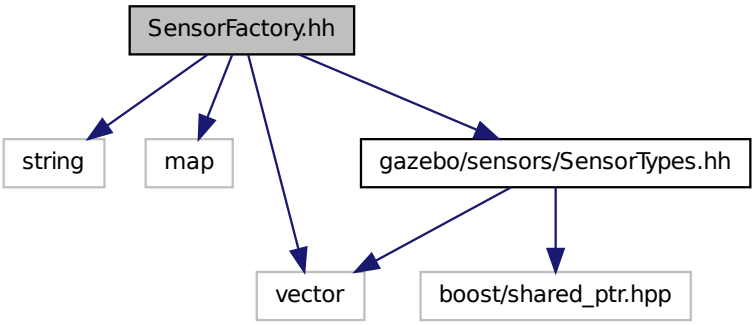
- enum **gazebo::sensors::SensorCategory** { **gazebo::sensors::IMAGE** = 0, **gazebo::sensors::RAY** = 1, **gazebo::sensors::OTHER** = 2, **gazebo::sensors::CATEGORY_COUNT** = 3 }

  *SensorClass is used to categorize sensors.*

# 11.159 SensorFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/sensors/SensorTypes.hh"
```

Include dependency graph for SensorFactory.hh:




### Classes

- class **gazebo::sensors::SensorFactory**

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

### Macros

- #define **GZ_REGISTER_STATIC_SENSOR**(name, classname)

  *Static sensor registration macro.*

### Typedefs

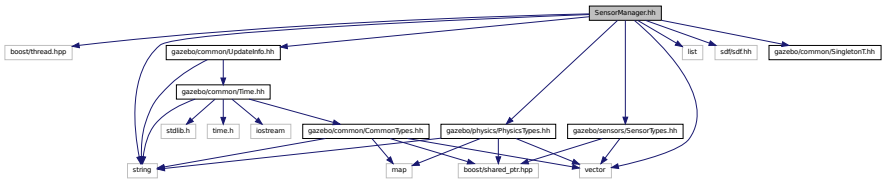
- typedef Sensor ∗(∗ **gazebo::sensors::SensorFactoryFn** )()

## 11.160 SensorManager.hh File Reference

```
#include <boost/thread.hpp>
```

```
#include <string>
#include <vector>
#include <list>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/sensors/SensorTypes.hh"
```

Include dependency graph for SensorManager.hh:




### Classes

- class **gazebo::sensors::SensorManager**

  *Class to manage and update all sensors.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

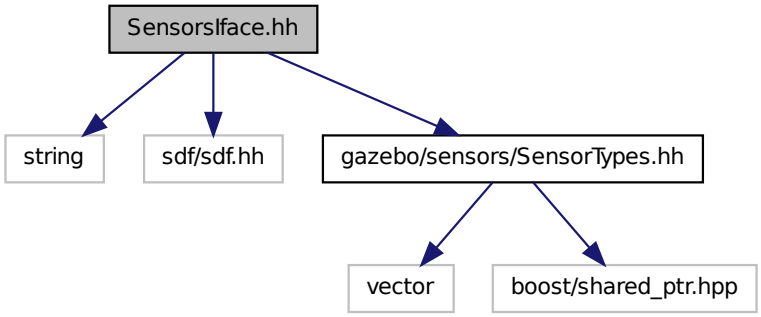
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.161 SensorsIface.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/sensors/SensorTypes.hh"
```

Include dependency graph for SensorsIface.hh:




## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## Functions

- std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(2.0)

  *Deprecated.*
- std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  *Create a sensor using SDF.*
- void **gazebo::sensors::disable** ()

  *Disable sensors.*
- void **gazebo::sensors::enable** ()

  *Enable sensors.*
- bool **gazebo::sensors::fini** ()

  *shutdown the sensor generation loop.*
- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)

  *Get a sensor using by name.*
- bool **gazebo::sensors::init** ()

  *initialize the sensor generation loop.*
- bool **gazebo::sensors::load** ()

  *Load the sensor library.*
- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)

  *Remove a sensor by name.*
- bool **gazebo::sensors::remove_sensors** ()

*Remove all sensors.*

- void **gazebo::sensors::run_once** (bool _force=false)

  *Run the sensor generation one step.*

- void **gazebo::sensors::run_threads** ()

  *Run sensors in a threads. This is a non-blocking call.*

- void **gazebo::sensors::stop** ()

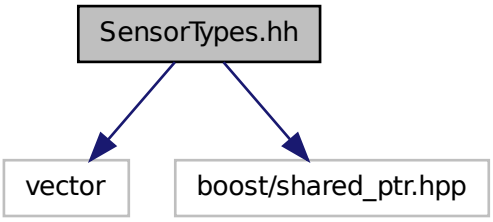
  *Stop the sensor generation loop.*

## 11.162 SensorTypes.hh File Reference

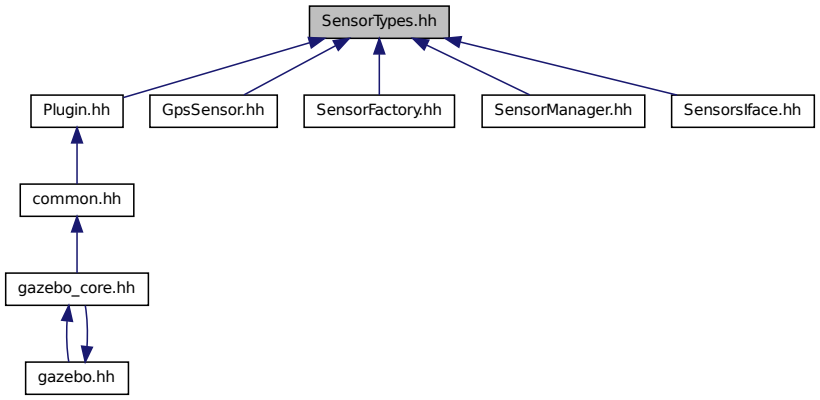
Forward declarations and typedefs for sensors.

```
#include <vector>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for SensorTypes.hh:




This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## Typedefs

- typedef std::vector < CameraSensorPtr > **gazebo::sensors::CameraSensor_V**
- typedef boost::shared_ptr < CameraSensor > **gazebo::sensors::CameraSensorPtr**
- typedef std::vector < ContactSensorPtr > **gazebo::sensors::ContactSensor_V**
- typedef boost::shared_ptr < ContactSensor > **gazebo::sensors::ContactSensorPtr**
- typedef std::vector < DepthCameraSensorPtr > **gazebo::sensors::DepthCameraSensor_V**
- typedef boost::shared_ptr < DepthCameraSensor > **gazebo::sensors::DepthCameraSensorPtr**
- typedef boost::shared_ptr < ForceTorqueSensor > **gazebo::sensors::ForceTorqueSensorPtr**
- typedef boost::shared_ptr < GpsSensor > **gazebo::sensors::GpsSensorPtr**
- typedef std::vector < GpuRaySensorPtr > **gazebo::sensors::GpuRaySensor_V**
- typedef boost::shared_ptr < GpuRaySensor > **gazebo::sensors::GpuRaySensorPtr**
- typedef std::vector< ImuSensorPtr > **gazebo::sensors::ImuSensor_V**
- typedef boost::shared_ptr < ImuSensor > **gazebo::sensors::ImuSensorPtr**
- typedef std::vector < MultiCameraSensorPtr > **gazebo::sensors::MultiCameraSensor_V**
- typedef boost::shared_ptr < MultiCameraSensor > **gazebo::sensors::MultiCameraSensorPtr**
- typedef boost::shared_ptr< Noise > **gazebo::sensors::NoisePtr**
- typedef std::vector< RaySensorPtr > **gazebo::sensors::RaySensor_V**
- typedef boost::shared_ptr < RaySensor > **gazebo::sensors::RaySensorPtr**
- typedef std::vector< RFIDSensor > **gazebo::sensors::RFIDSensor_V**
- typedef boost::shared_ptr < RFIDSensor > **gazebo::sensors::RFIDSensorPtr**
- typedef std::vector< RFIDTag > **gazebo::sensors::RFIDTag_V**
- typedef boost::shared_ptr < RFIDTag > **gazebo::sensors::RFIDTagPtr**
- typedef std::vector< SensorPtr > **gazebo::sensors::Sensor_V**
- typedef boost::shared_ptr< Sensor > **gazebo::sensors::SensorPtr**
- typedef boost::shared_ptr < SonarSensor > **gazebo::sensors::SonarSensorPtr**

- typedef std::vector
  < WirelessReceiver > **gazebo::sensors::WirelessReceiver_V**

- typedef boost::shared_ptr
  < WirelessReceiver > **gazebo::sensors::WirelessReceiverPtr**

- typedef std::vector
  < WirelessTransceiver > **gazebo::sensors::WirelessTransceiver_V**

- typedef boost::shared_ptr
  < WirelessTransceiver > **gazebo::sensors::WirelessTransceiverPtr**

- typedef std::vector
  < WirelessTransmitter > **gazebo::sensors::WirelessTransmitter_V**

- typedef boost::shared_ptr
  < WirelessTransmitter > **gazebo::sensors::WirelessTransmitterPtr**

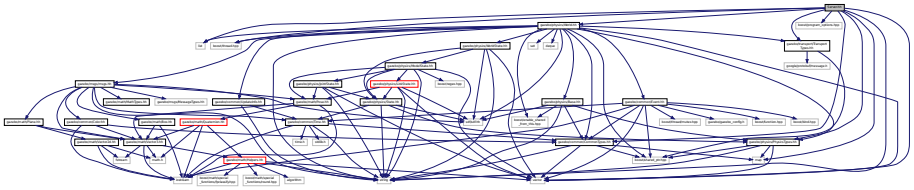
### 11.162.1 Detailed Description

Forward declarations and typedefs for sensors.

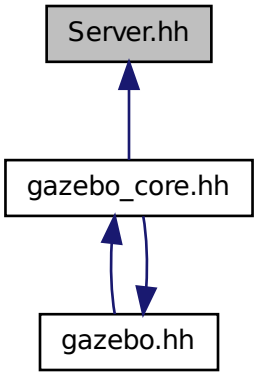
## 11.163 Server.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include <map>
#include <boost/program_options.hpp>
#include <boost/thread.hpp>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/World.hh"
```

Include dependency graph for Server.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::Server**

### Namespaces

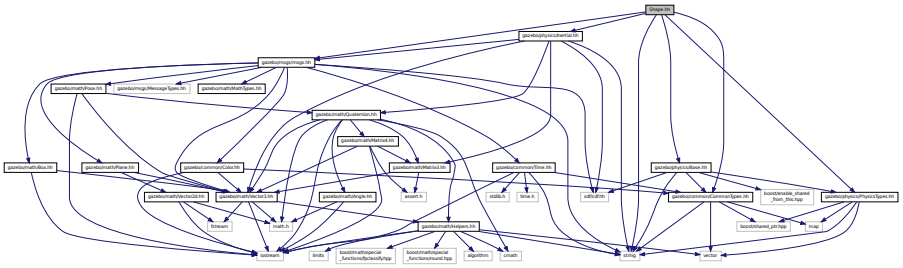
- namespace **boost**
- namespace **gazebo**

  *Forward declarations for the common classes.*

## 11.164 Shape.hh File Reference

```
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Inertial.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Shape.hh:



This graph shows which files directly or indirectly include this file:

## Classes

- class **gazebo::physics::Shape**

  ***Base*** *(*p. *159) class for all shapes.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

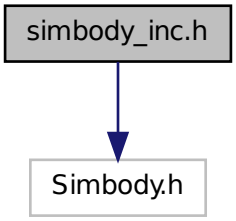
- namespace **gazebo::physics**

  *namespace for physics*

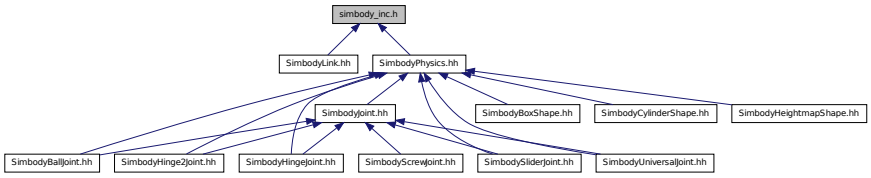
# 11.165 simbody_inc.h File Reference

```
#include <Simbody.h>
```

Include dependency graph for simbody_inc.h:



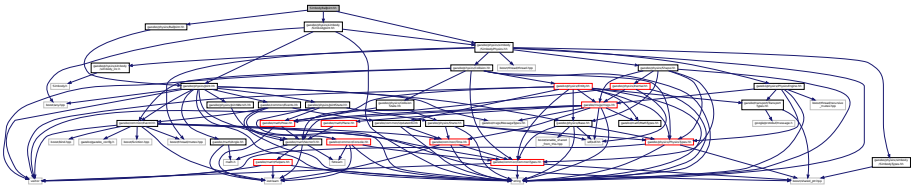

This graph shows which files directly or indirectly include this file:




## 11.166 SimbodyBallJoint.hh File Reference

```
#include "gazebo/physics/BallJoint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
```

Include dependency graph for SimbodyBallJoint.hh:



### Classes

- class **gazebo::physics::SimbodyBallJoint**

  ***SimbodyBallJoint*** *(p. 865) class models a ball joint in Simbody.*

### Namespaces

- namespace **gazebo**

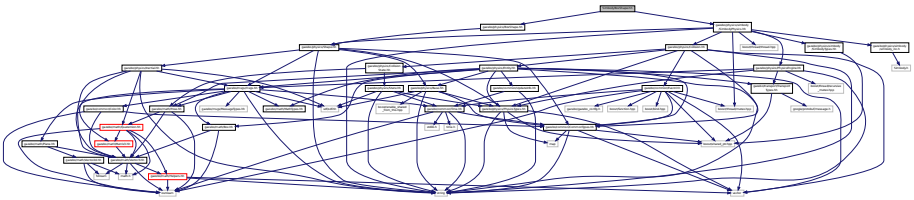
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.167 SimbodyBoxShape.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/BoxShape.hh"
```

Include dependency graph for SimbodyBoxShape.hh:



### Classes

- class **gazebo::physics::SimbodyBoxShape**

  *Simbody box collision.*

### Namespaces

- namespace **gazebo**

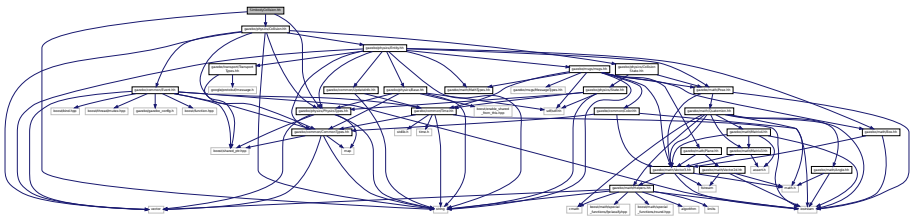
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.168 SimbodyCollision.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Collision.hh"
```

Include dependency graph for SimbodyCollision.hh:



### Classes

- class **gazebo::physics::SimbodyCollision**

  *Simbody collisions.*

### Namespaces

- namespace **gazebo**

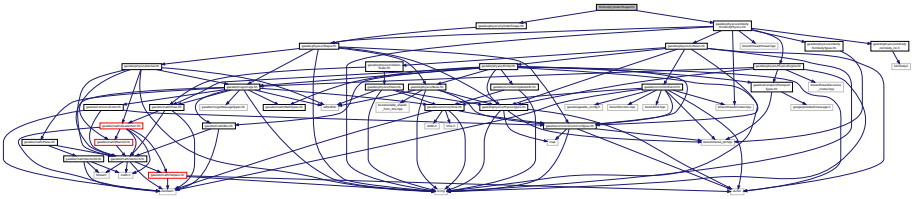
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*
- namespace **SimTK**

## 11.169 SimbodyCylinderShape.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/CylinderShape.hh"
```

Include dependency graph for SimbodyCylinderShape.hh:



### Classes

- class **gazebo::physics::SimbodyCylinderShape**

  *Cylinder collision.*

### Namespaces

- namespace **gazebo**

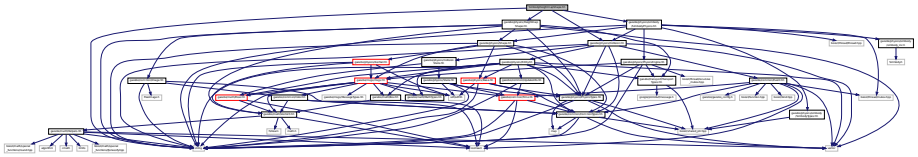
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.170 SimbodyHeightmapShape.hh File Reference

```
#include <string>
#include "gazebo/physics/HeightmapShape.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/Collision.hh"
```

Include dependency graph for SimbodyHeightmapShape.hh:



### Classes

- class **gazebo::physics::SimbodyHeightmapShape**

  *Height map collision.*

### Namespaces

- namespace **gazebo**

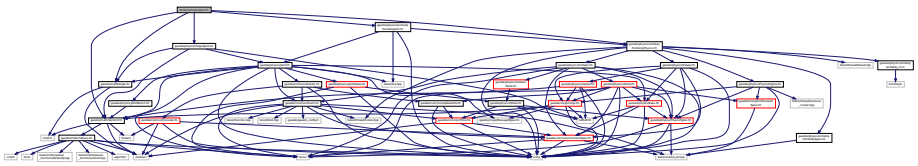
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.171 SimbodyHinge2Joint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Hinge2Joint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
```

Include dependency graph for SimbodyHinge2Joint.hh:



### Classes

- class **gazebo::physics::SimbodyHinge2Joint**

  *A two axis hinge joint.*

### Namespaces

- namespace **gazebo**

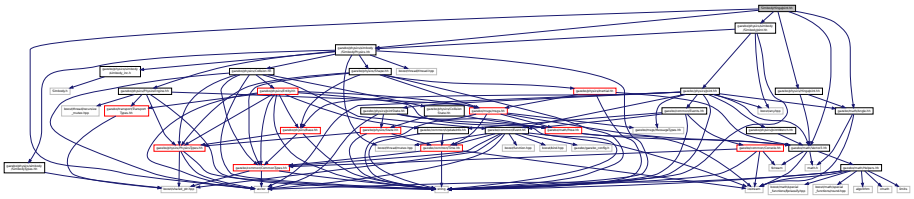
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.172 SimbodyHingeJoint.hh File Reference

```
#include <vector>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/HingeJoint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
```

Include dependency graph for SimbodyHingeJoint.hh:



### Classes

- class **gazebo::physics::SimbodyHingeJoint**

  *A single axis hinge joint.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

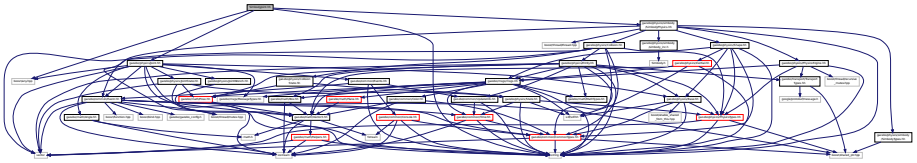
- namespace **gazebo::physics**

  *namespace for physics*

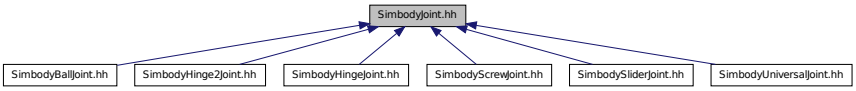
## 11.173 SimbodyJoint.hh File Reference

```
#include <boost/any.hpp>
#include <string>
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for SimbodyJoint.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::SimbodyJoint**

  ***Base*** (p. *159*) *class for all joints.*

### Namespaces

- namespace **gazebo**

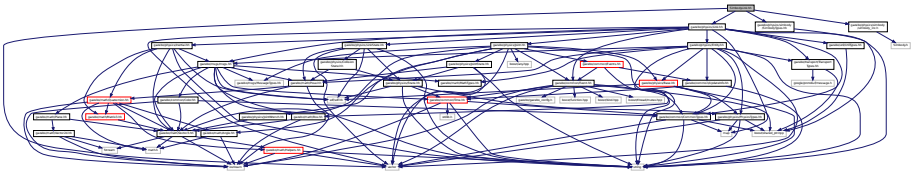
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.174 SimbodyLink.hh File Reference

```
#include <vector>
#include "gazebo/physics/simbody/SimbodyTypes.hh"
#include "gazebo/physics/Link.hh"
#include "gazebo/physics/simbody/simbody_inc.h"
```

Include dependency graph for SimbodyLink.hh:



### Classes

- class **gazebo::physics::SimbodyLink**

  *Simbody* ***Link*** (p. *542*) *class.*

### Namespaces

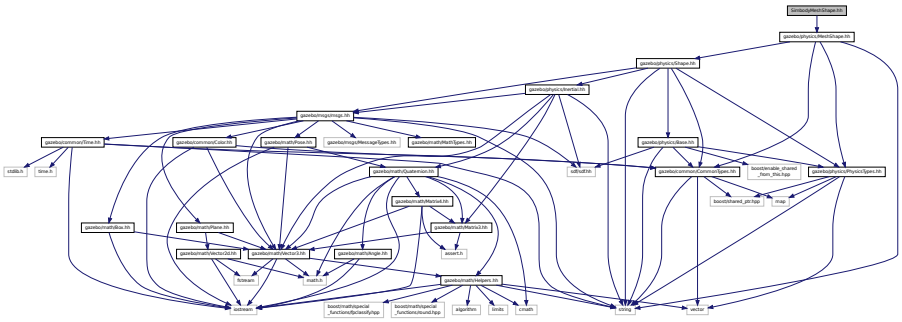
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.175 SimbodyMeshShape.hh File Reference

`#include "gazebo/physics/MeshShape.hh"`
Include dependency graph for SimbodyMeshShape.hh:



### Classes

- class **gazebo::physics::SimbodyMeshShape**

  *Triangle mesh collision.*

### Namespaces

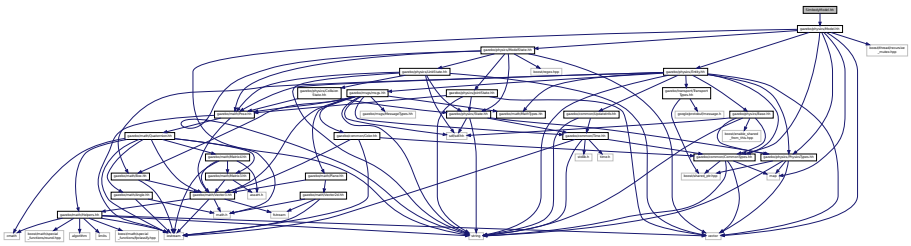
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.176 SimbodyModel.hh File Reference

`#include "gazebo/physics/Model.hh"`
Include dependency graph for SimbodyModel.hh:



### Classes

- class **gazebo::physics::SimbodyModel**

  *A model is a collection of links, joints, and plugins.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

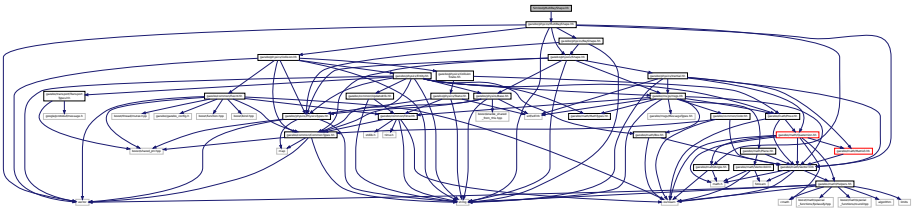
- namespace **gazebo::physics**

  *namespace for physics*

## 11.177 SimbodyMultiRayShape.hh File Reference

```
#include "gazebo/physics/MultiRayShape.hh"
```

Include dependency graph for SimbodyMultiRayShape.hh:



### Classes

- class **gazebo::physics::SimbodyMultiRayShape**

  *Simbody specific version of* ***MultiRayShape*** (p. *664*).

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

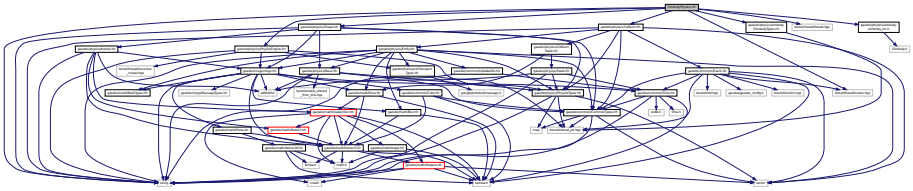
- namespace **gazebo::physics**

  *namespace for physics*

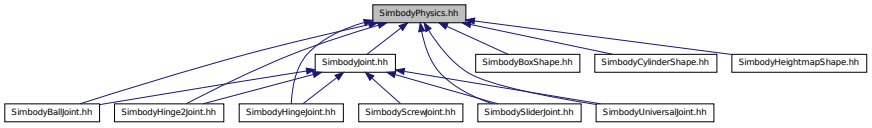
## 11.178 SimbodyPhysics.hh File Reference

```
#include <string>
#include <boost/thread/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "gazebo/physics/PhysicsEngine.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/simbody/SimbodyTypes.hh"
#include "gazebo/physics/simbody/simbody_inc.h"
```

Include dependency graph for SimbodyPhysics.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::SimbodyPhysics**

  *Simbody physics engine.*

### Namespaces

- namespace **gazebo**

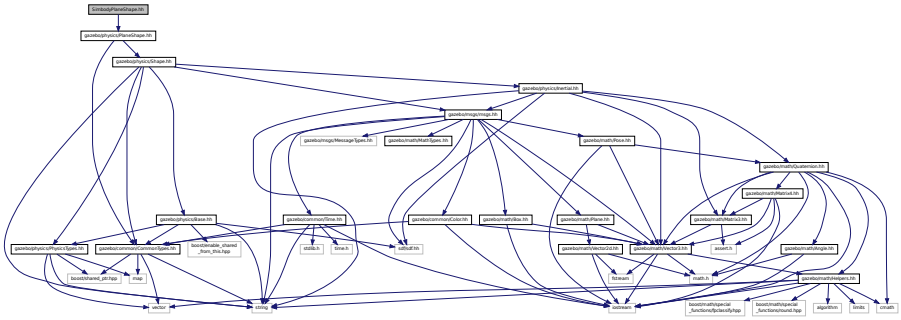
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.179 SimbodyPlaneShape.hh File Reference

```
#include "gazebo/physics/PlaneShape.hh"
```

Include dependency graph for SimbodyPlaneShape.hh:



### Classes

- class **gazebo::physics::SimbodyPlaneShape**

  *Simbody collision for an infinite plane.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

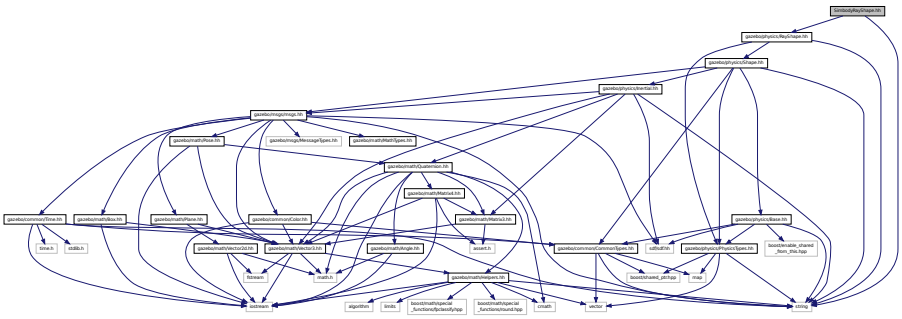
- namespace **gazebo::physics**

  *namespace for physics*

## 11.180 SimbodyRayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/RayShape.hh"
```

Include dependency graph for SimbodyRayShape.hh:



### Classes

- class **gazebo::physics::SimbodyRayShape**

  *Ray shape for simbody.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

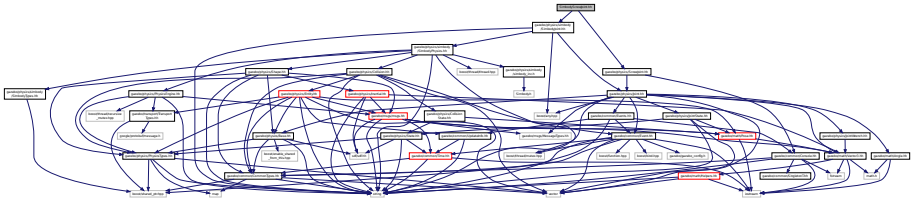
- namespace **gazebo::physics**

  *namespace for physics*

## 11.181 SimbodyScrewJoint.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/ScrewJoint.hh"
```

Include dependency graph for SimbodyScrewJoint.hh:



### Classes

- class **gazebo::physics::SimbodyScrewJoint**

  *A screw joint.*

### Namespaces

- namespace **gazebo**

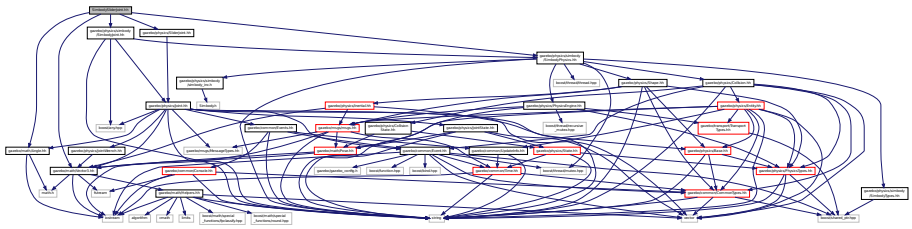
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.182 SimbodySliderJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/SliderJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
```

Include dependency graph for SimbodySliderJoint.hh:



### Classes

- class **gazebo::physics::SimbodySliderJoint**

  *A slider joint.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

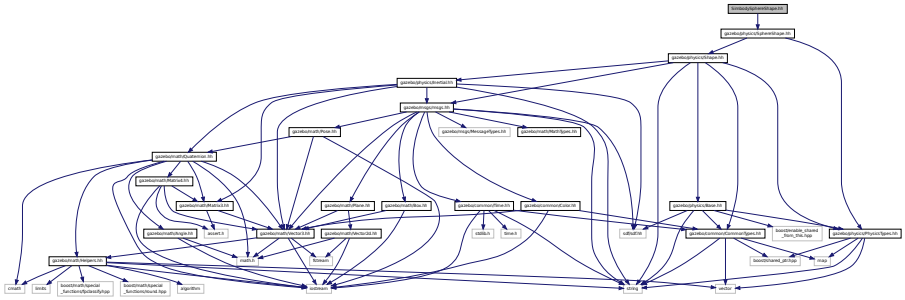
- namespace **gazebo::physics**

  *namespace for physics*

## 11.183 SimbodySphereShape.hh File Reference

```
#include "gazebo/physics/SphereShape.hh"
```

Include dependency graph for SimbodySphereShape.hh:



### Classes

- class **gazebo::physics::SimbodySphereShape**

  *Simbody sphere collision.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::physics**

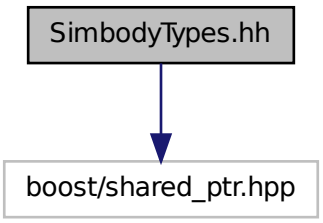
  *namespace for physics*

## 11.184 SimbodyTypes.hh File Reference

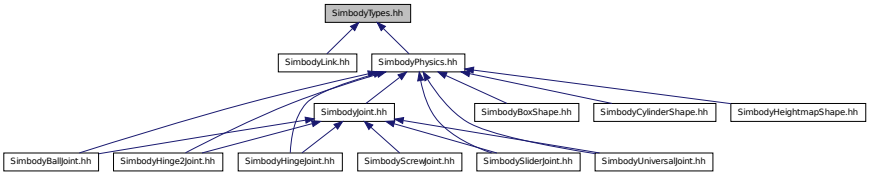
Simbody wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for SimbodyTypes.hh:




This graph shows which files directly or indirectly include this file:




## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## Typedefs

- typedef boost::shared_ptr < SimbodyCollision > **gazebo::physics::SimbodyCollisionPtr**
- typedef boost::shared_ptr < SimbodyLink > **gazebo::physics::SimbodyLinkPtr**
- typedef boost::shared_ptr < SimbodyModel > **gazebo::physics::SimbodyModelPtr**
- typedef boost::shared_ptr < SimbodyPhysics > **gazebo::physics::SimbodyPhysicsPtr**
- typedef boost::shared_ptr < SimbodyRayShape > **gazebo::physics::SimbodyRayShapePtr**

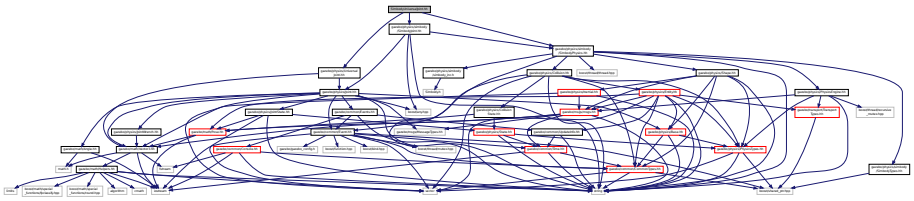
### 11.184.1 Detailed Description

Simbody wrapper forward declarations and typedefs.

## 11.185 SimbodyUniversalJoint.hh File Reference

```
#include "gazebo/physics/UniversalJoint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
```

Include dependency graph for SimbodyUniversalJoint.hh:



### Classes

- class **gazebo::physics::SimbodyUniversalJoint**

  *A simbody universal joint class.*

### Namespaces

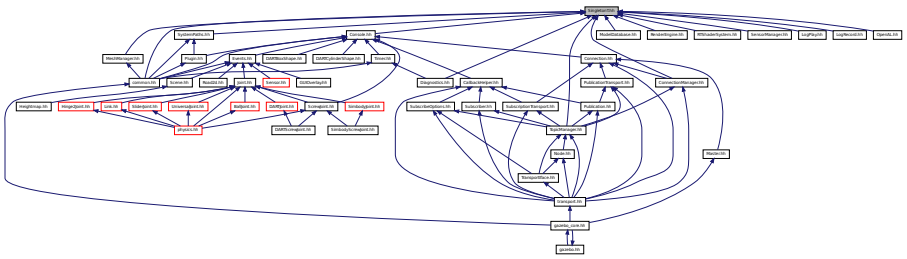
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.186 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:



### Classes

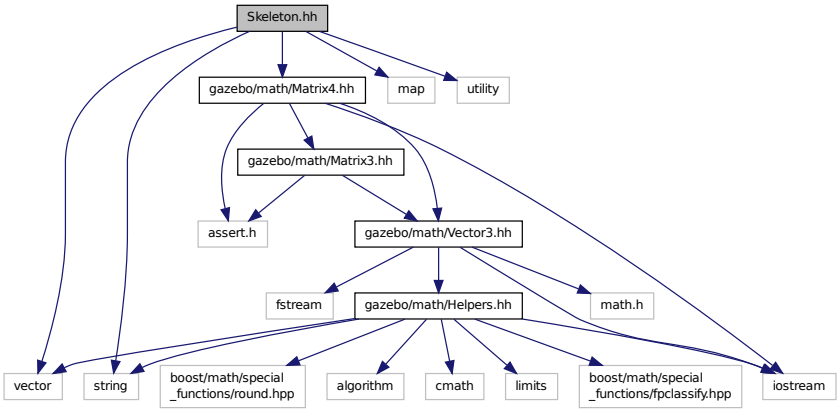
- class **SingletonT< T >**

  *Singleton template class.*

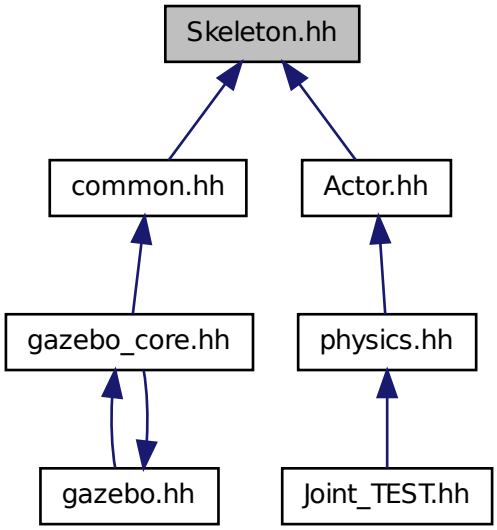
## 11.187 Skeleton.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <utility>
#include "gazebo/math/Matrix4.hh"
```

Include dependency graph for Skeleton.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::common::NodeTransform**

  ***NodeTransform*** *(*p. *684)* ***Skeleton.hh*** *(*p. *1367) common/common.hh*
- class **gazebo::common::Skeleton**

  *A skeleton.*
- class **gazebo::common::SkeletonNode**

  *A skeleton node.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## Typedefs

- typedef std::map< unsigned int, SkeletonNode * > **gazebo::common::NodeMap**
- typedef std::map< unsigned int, SkeletonNode * >::iterator **gazebo::common::NodeMapIter**

- typedef std::map< double, std::vector< NodeTransform > > **gazebo::common::RawNodeAnim**

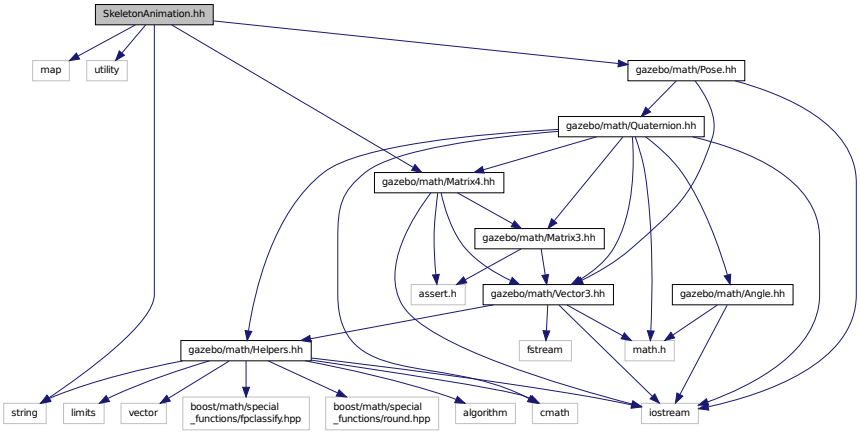
- typedef std::vector < std::vector< std::pair < std::string, double > > > **gazebo::common::RawNodeWeights**

- typedef std::map< std::string, RawNodeAnim > **gazebo::common::RawSkeletonAnim**

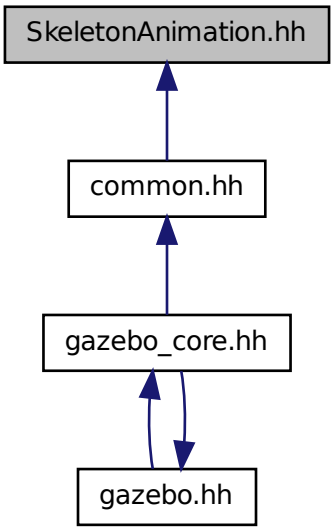
## 11.188 SkeletonAnimation.hh File Reference

```
#include <map>
#include <utility>
#include <string>
#include "gazebo/math/Matrix4.hh"
#include "gazebo/math/Pose.hh"
```

Include dependency graph for SkeletonAnimation.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::NodeAnimation**

  *Node animation.*

- class **gazebo::common::SkeletonAnimation**

  ***Skeleton*** *(p. 953) animation.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

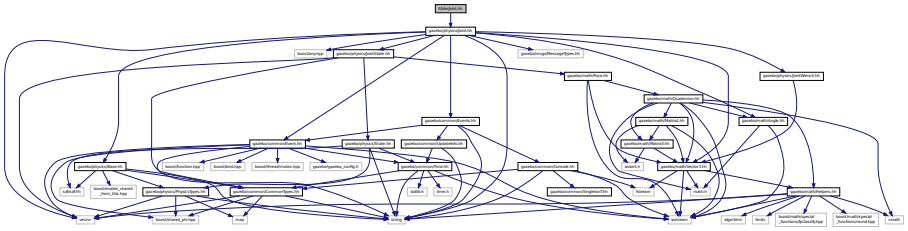
- namespace **gazebo::common**

  *Common namespace.*

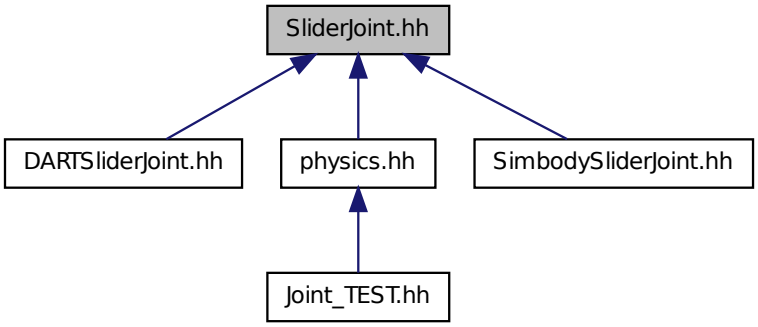
## 11.189 SliderJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for SliderJoint.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::SliderJoint**< **T** >

  *A slider joint.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

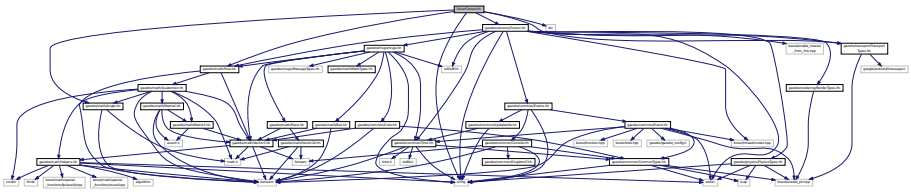
  *namespace for physics*

## 11.190 SonarSensor.hh File Reference

```
#include <string>
```

```
#include <list>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for SonarSensor.hh:



### Classes

- class **gazebo::sensors::SonarSensor**

  ***Sensor*** *(p. 837) with sonar cone.*

### Namespaces

- namespace **gazebo**

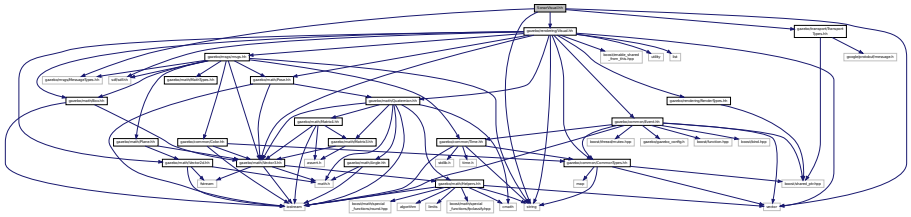
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.191 SonarVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for SonarVisual.hh:



### Classes

- class **gazebo::rendering::SonarVisual**

  *Visualization for sonar data.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

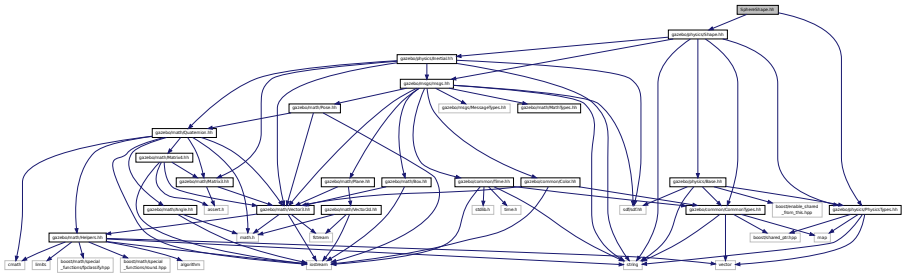
- namespace **gazebo::rendering**

  *Rendering namespace.*

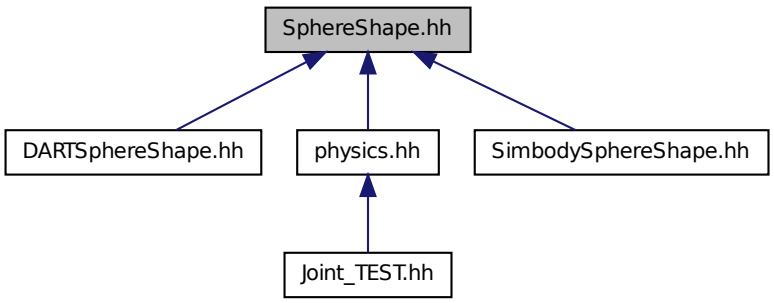
## 11.192 SphereShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for SphereShape.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::SphereShape**

  *Sphere collision shape.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

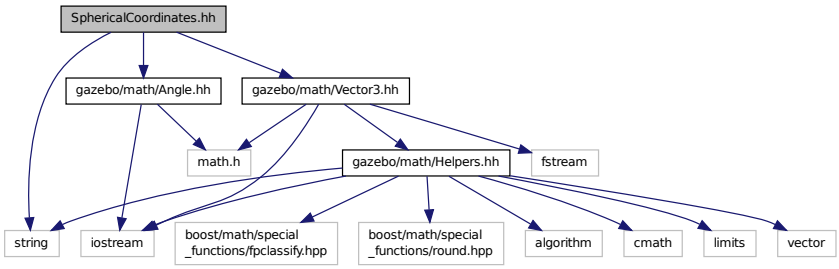
- namespace **gazebo::physics**

  *namespace for physics*

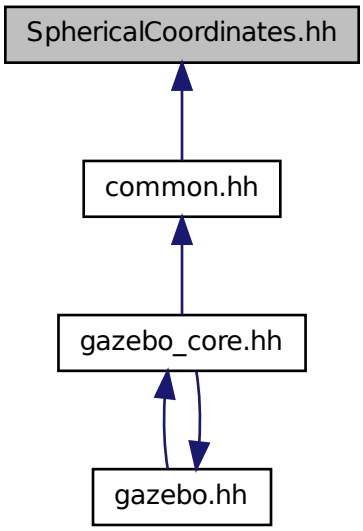
## 11.193 SphericalCoordinates.hh File Reference

```
#include <string>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for SphericalCoordinates.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::SphericalCoordinates**

  *Convert spherical coordinates for planetary surfaces.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

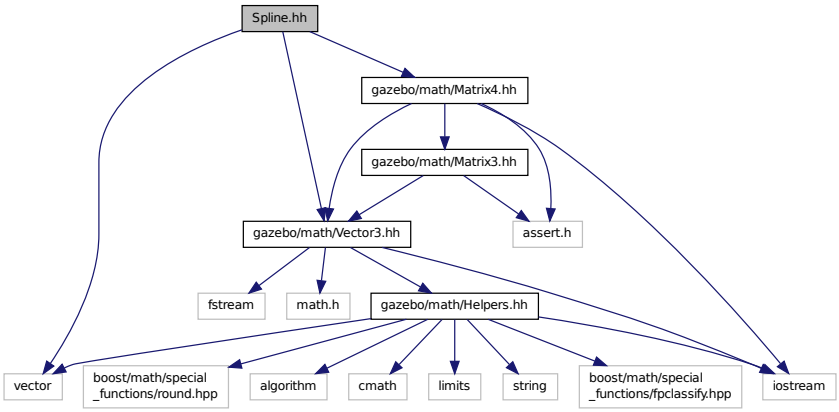
- namespace **gazebo::common**

  *Common namespace.*

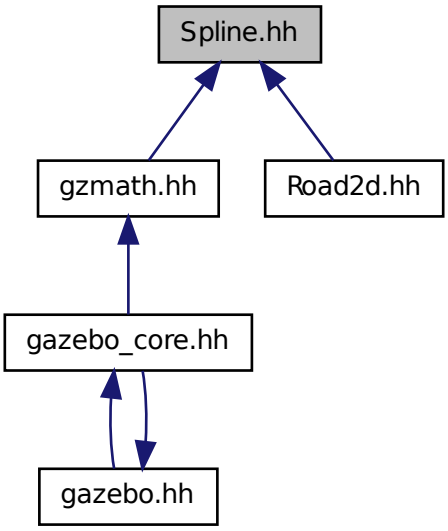
## 11.194 Spline.hh File Reference

```
#include <vector>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix4.hh"
```

Include dependency graph for Spline.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::math::Spline**

  *Splines.*

### Namespaces

- namespace **gazebo**

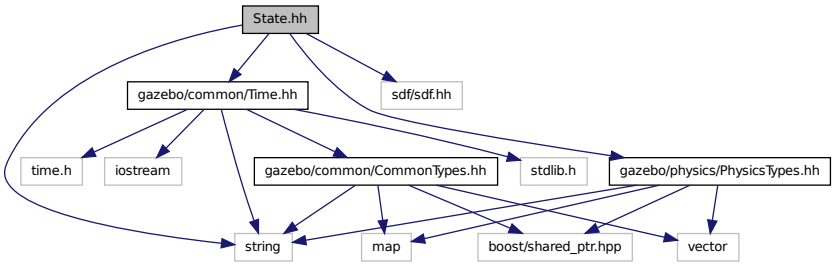
  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

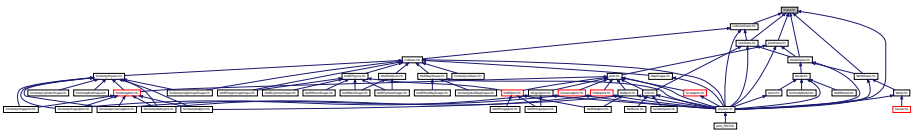
## 11.195 State.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/Time.hh"
```

Include dependency graph for State.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::physics::State**

  ***State*** *(p. 998) of an entity.*

### Namespaces

- namespace **gazebo**

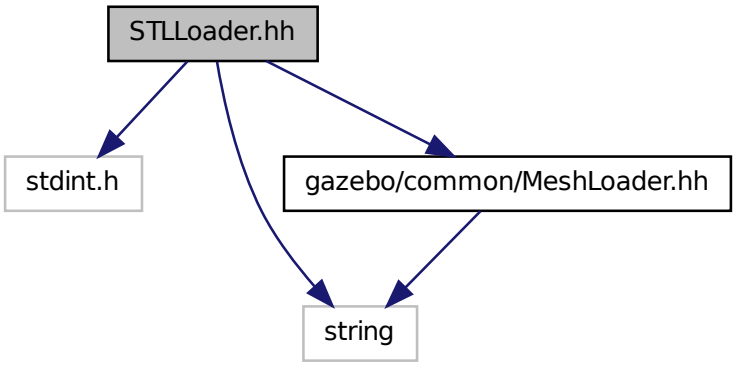
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

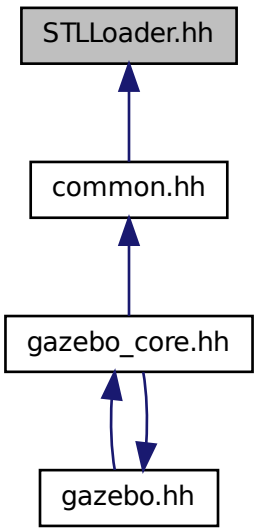
## 11.196 STLLoader.hh File Reference

```
#include <stdint.h>
#include <string>
#include "gazebo/common/MeshLoader.hh"
```

Include dependency graph for STLLoader.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::STLLoader**

  *Class used to load STL mesh files.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

### Macros

- #define **COR3_MAX** 200000
- #define **FACE_MAX** 200000
- #define **LINE_MAX_LEN** 256
- #define **ORDER_MAX** 10

#### 11.196.1 Macro Definition Documentation

##### 11.196.1.1 #define COR3_MAX 200000

##### 11.196.1.2 #define FACE_MAX 200000

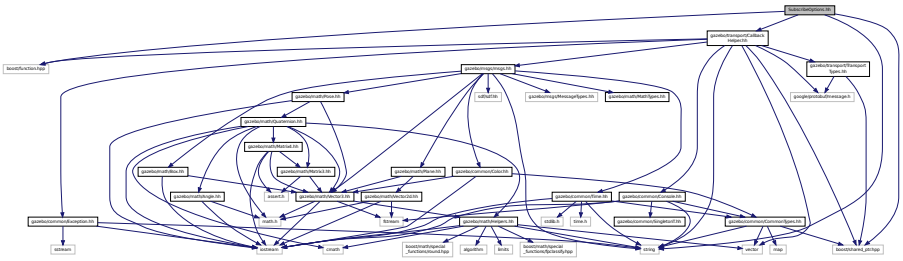
##### 11.196.1.3 #define LINE_MAX_LEN 256

##### 11.196.1.4 #define ORDER_MAX 10

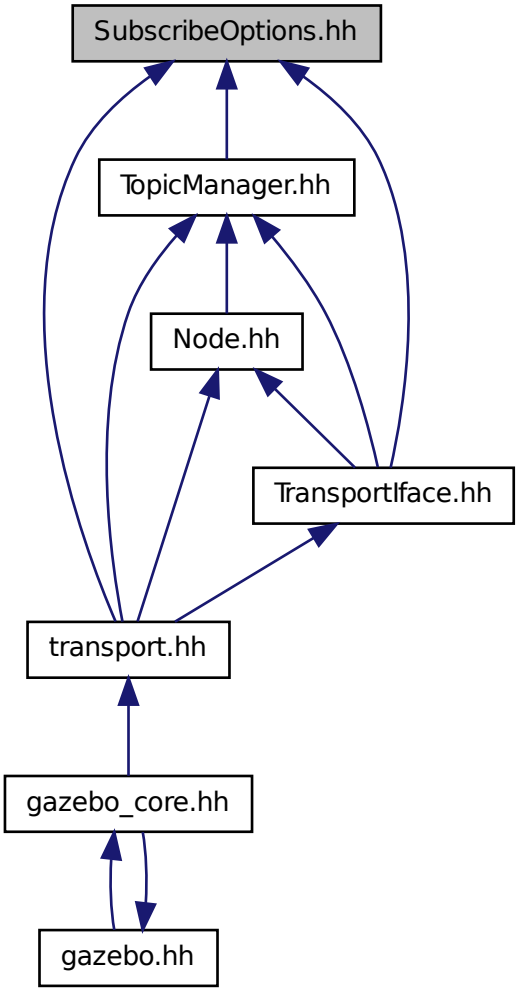
## 11.197 SubscribeOptions.hh File Reference

```
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <string>
#include "gazebo/transport/CallbackHelper.hh"
```

Include dependency graph for SubscribeOptions.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::SubscribeOptions**

  *Options for a subscription.*

## Namespaces

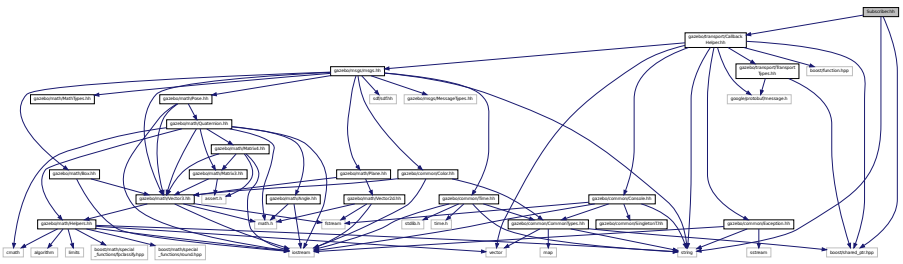
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

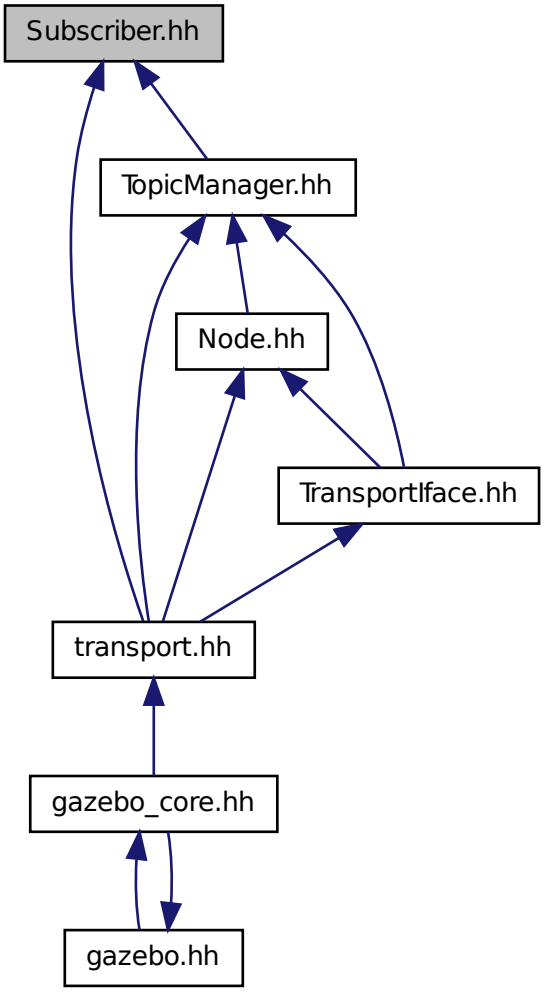
## 11.198 Subscriber.hh File Reference

```
#include <string>
#include <boost/shared_ptr.hpp>
#include "gazebo/transport/CallbackHelper.hh"
```

Include dependency graph for Subscriber.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::Subscriber**

  *A subscriber to a topic.*

## Namespaces

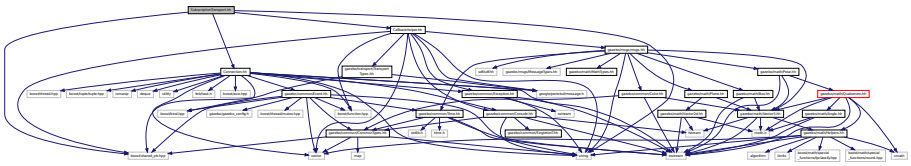
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

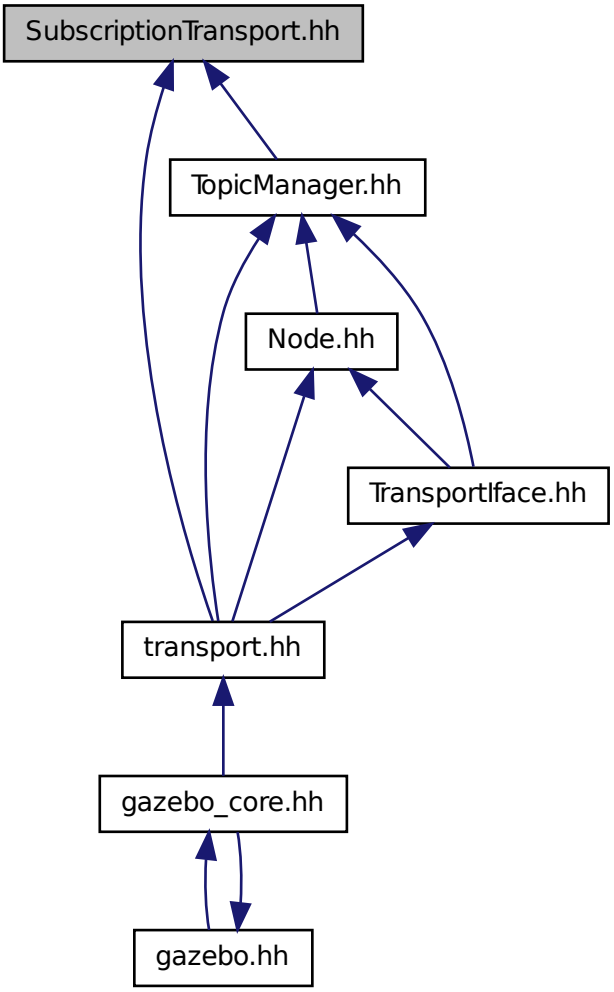
## 11.199 SubscriptionTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "Connection.hh"
#include "CallbackHelper.hh"
```

Include dependency graph for SubscriptionTransport.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::transport::SubscriptionTransport**

  *transport/transport.hh*

## Namespaces

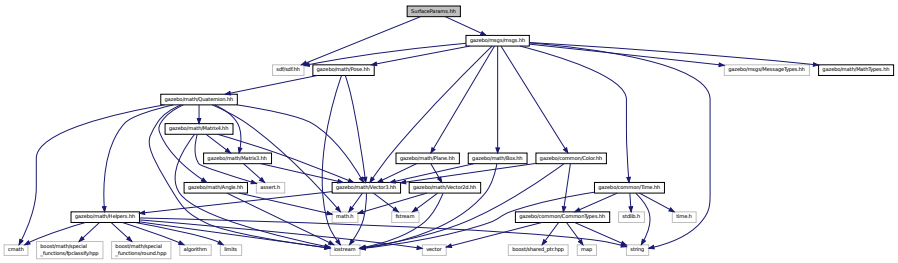
- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

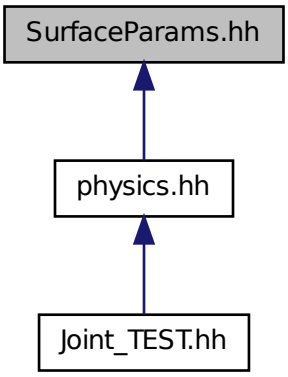
## 11.200 SurfaceParams.hh File Reference

```
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
```

Include dependency graph for SurfaceParams.hh:




This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::SurfaceParams**

  ***SurfaceParams*** (p. *1020) defines various Surface contact parameters.*

### Namespaces

- namespace **gazebo**

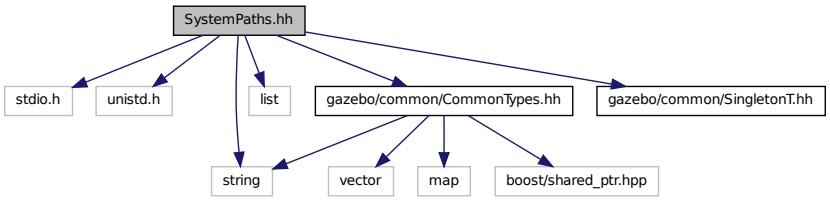
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

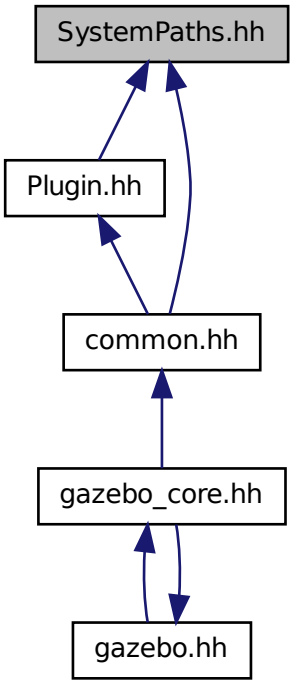
## 11.201 SystemPaths.hh File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <string>
#include <list>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for SystemPaths.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::common::SystemPaths**

  *Functions to handle getting system paths, keeps track of:*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

### Macros

- #define **GetCurrentDir** getcwd
- #define **LINUX**

### 11.201.1 Macro Definition Documentation

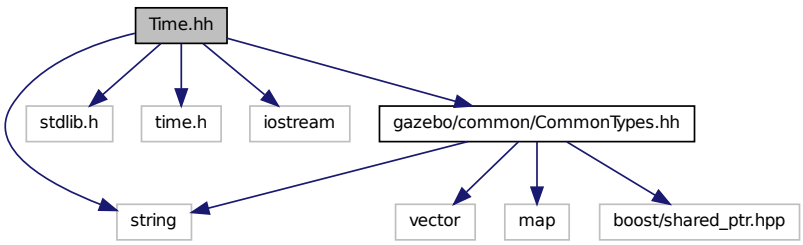
#### 11.201.1.1 #define GetCurrentDir getcwd

#### 11.201.1.2 #define LINUX

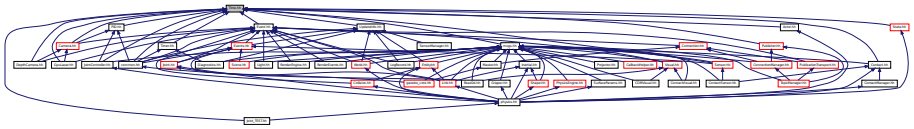
## 11.202 Time.hh File Reference

```
#include <string>
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Time.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::common::Time**

  *A **Time** (p. 1031) class, can be used to hold wall- or sim-time.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

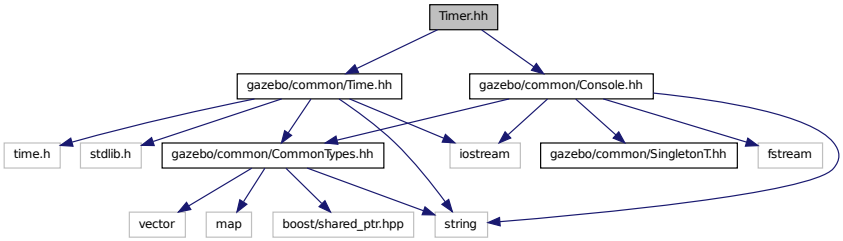
- namespace **gazebo::common**

  *Common namespace.*

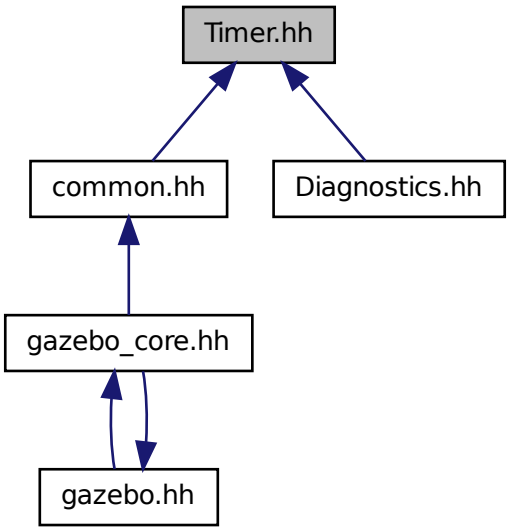
# 11.203 Timer.hh File Reference

```
#include "gazebo/common/Console.hh"
#include "gazebo/common/Time.hh"
```

Include dependency graph for Timer.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::Timer**

  *A timer class, used to time things in real world walltime.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::common**

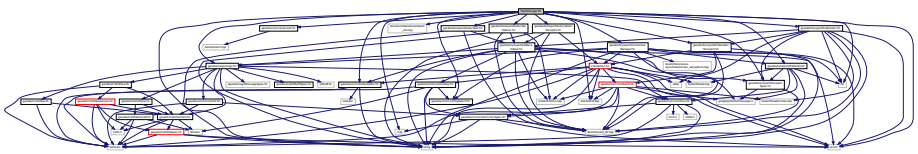
  *Common namespace.*

## 11.204 TopicManager.hh File Reference

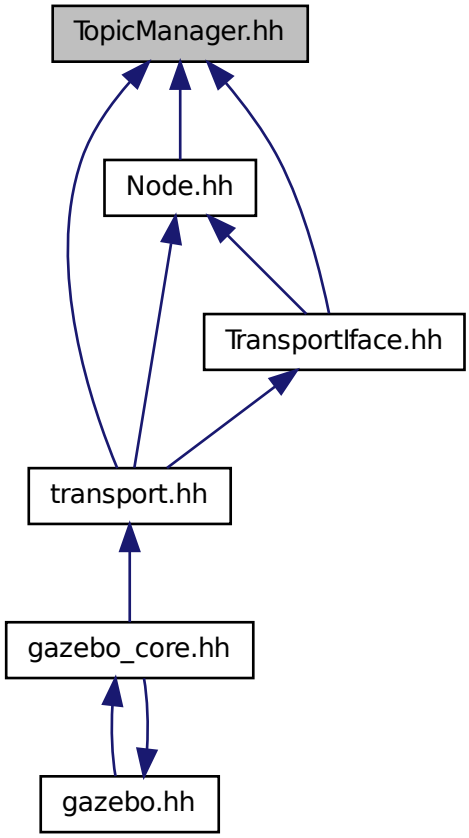
```
#include <boost/bind.hpp>
```

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <boost/unordered/unordered_set.hpp>
#include "gazebo/common/Assert.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/SubscribeOptions.hh"
#include "gazebo/transport/SubscriptionTransport.hh"
#include "gazebo/transport/PublicationTransport.hh"
#include "gazebo/transport/ConnectionManager.hh"
#include "gazebo/transport/Publisher.hh"
#include "gazebo/transport/Publication.hh"
#include "gazebo/transport/Subscriber.hh"
```

Include dependency graph for TopicManager.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::transport::TopicManager**

  *Manages topics and their subscriptions.*

### Namespaces

- namespace **gazebo**

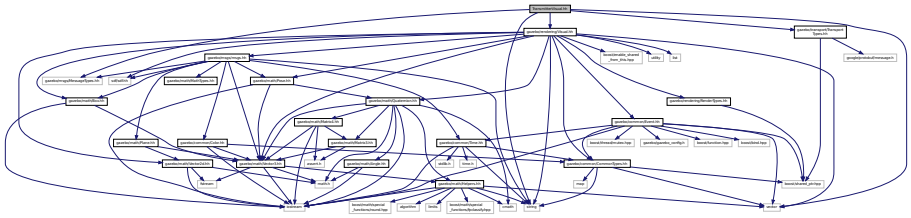
  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

## 11.205 TransmitterVisual.hh File Reference

```
#include <string>
```

```
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for TransmitterVisual.hh:



### Classes

- class **gazebo::rendering::TransmitterVisual**

  *Visualization for the wireless propagation data.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

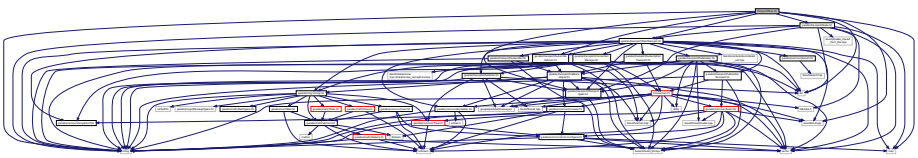
- namespace **gazebo::rendering**

  *Rendering namespace.*

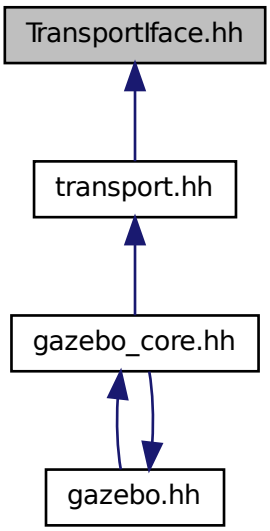
## 11.206 TransportIface.hh File Reference

```
#include <boost/bind.hpp>
#include <string>
#include <list>
#include <map>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/SubscribeOptions.hh"
#include "gazebo/transport/Node.hh"
#include "gazebo/transport/TopicManager.hh"
```

Include dependency graph for TransportIface.hh:



This graph shows which files directly or indirectly include this file:




## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

## Functions

- void **gazebo::transport::clear_buffers** ()

  *Clear any remaining communication buffers.*
- void **gazebo::transport::fini** ()

  *Cleanup the transport component.*
- bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)

  *Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.*
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)

  *Return all the namespace (world names) on the master.*
- std::map< std::string,
  std::list< std::string > > **gazebo::transport::getAdvertisedTopics** ()

  *Get a list of all the topics and their message types.*
- std::list< std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)

  *Get a list of all the unique advertised topic names.*
- bool **gazebo::transport::getMinimalComms** ()

  *Get whether minimal comms has been enabled.*

- std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)

  *Get the message typename that is published on the given topic.*
- bool **gazebo::transport::init** (const std::string &_master_host="", unsigned int _master_port=0)

  *Initialize the transport system.*
- bool **gazebo::transport::is_stopped** ()

  *Is the transport system stopped?*
- void **gazebo::transport::pause_incoming** (bool _pause)

  *Pause or unpause incoming messages.*
- template<typename M >
  void **gazebo::transport::publish** (const std::string &_topic, const google::protobuf::Message &_message)

  *A convenience function for a one-time publication of a message.*
- boost::shared_ptr< msgs::Response > **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  *Send a request and receive a response.*
- void **gazebo::transport::requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  *Send a request and don't wait for a response.*
- void **gazebo::transport::requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")

  *Send a request and don't wait for a response.*
- void **gazebo::transport::run** ()

  *Run the transport component.*
- void **gazebo::transport::setMinimalComms** (bool _enabled)

  *Set whether minimal comms should be used.*
- void **gazebo::transport::stop** ()

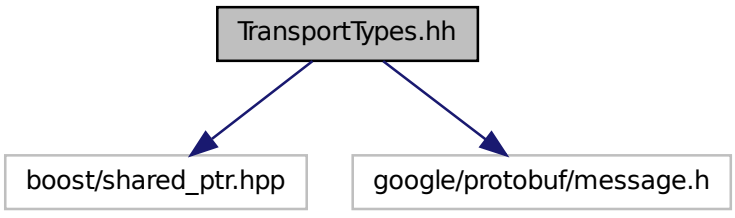
  *Stop the transport component from running.*

## 11.207 TransportTypes.hh File Reference

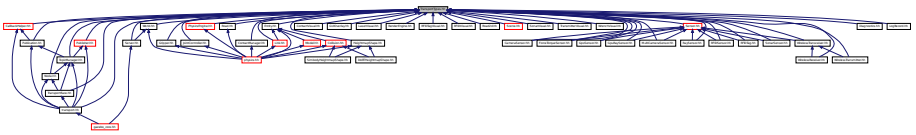
Forward declarations for transport.

```
#include <boost/shared_ptr.hpp>
#include <google/protobuf/message.h>
```

Include dependency graph for TransportTypes.hh:



This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::transport**

### Typedefs

- typedef boost::shared_ptr < google::protobuf::Message > **gazebo::transport::MessagePtr**
- typedef boost::shared_ptr< Node > **gazebo::transport::NodePtr**
- typedef boost::shared_ptr < Publication > **gazebo::transport::PublicationPtr**
- typedef boost::shared_ptr < PublicationTransport > **gazebo::transport::PublicationTransportPtr**
- typedef boost::shared_ptr < Publisher > **gazebo::transport::PublisherPtr**
- typedef boost::shared_ptr < Subscriber > **gazebo::transport::SubscriberPtr**
- typedef boost::shared_ptr < SubscriptionTransport > **gazebo::transport::SubscriptionTransportPtr**

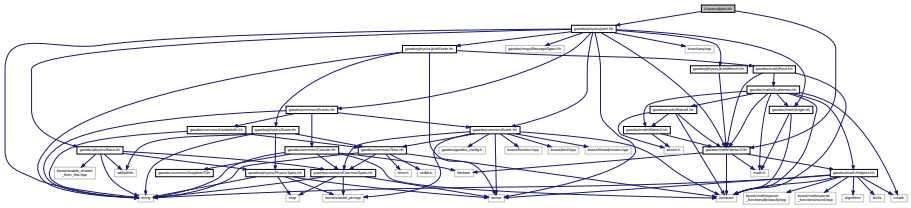
#### 11.207.1 Detailed Description

Forward declarations for transport.

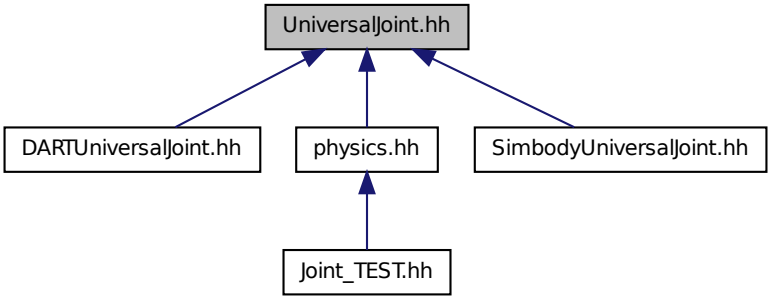
## 11.208 UniversalJoint.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for UniversalJoint.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::UniversalJoint**< **T** >
  *A universal joint.*

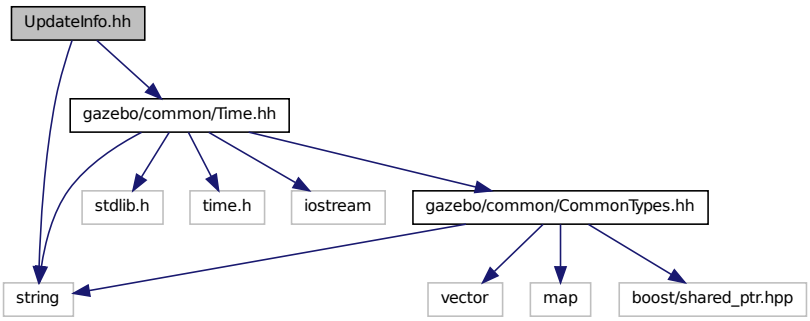
### Namespaces

- namespace **gazebo**
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**
  *namespace for physics*

## 11.209 UpdateInfo.hh File Reference

```
#include <string>
#include "gazebo/common/Time.hh"
```

Include dependency graph for UpdateInfo.hh:



This graph shows which files directly or indirectly include this file:

### Classes

- class **gazebo::common::UpdateInfo**

  *Information for use in an update event.*

### Namespaces

- namespace **gazebo**

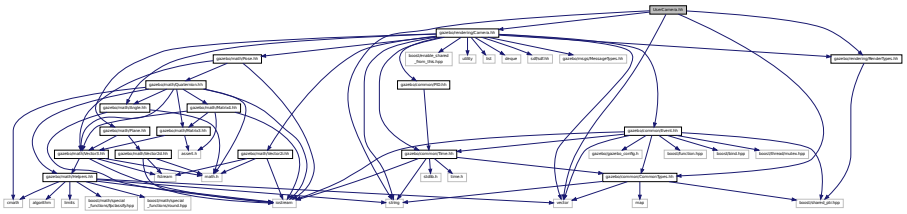
  *Forward declarations for the common classes.*
- namespace **gazebo::common**

  *Common namespace.*

## 11.210 UserCamera.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Camera.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for UserCamera.hh:



### Classes

- class **gazebo::rendering::UserCamera**

  *A camera used for user visualization of a scene.*

### Namespaces

- namespace **gazebo**

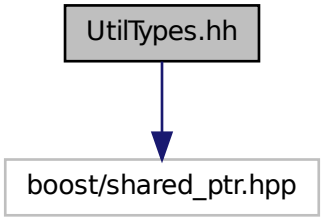
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

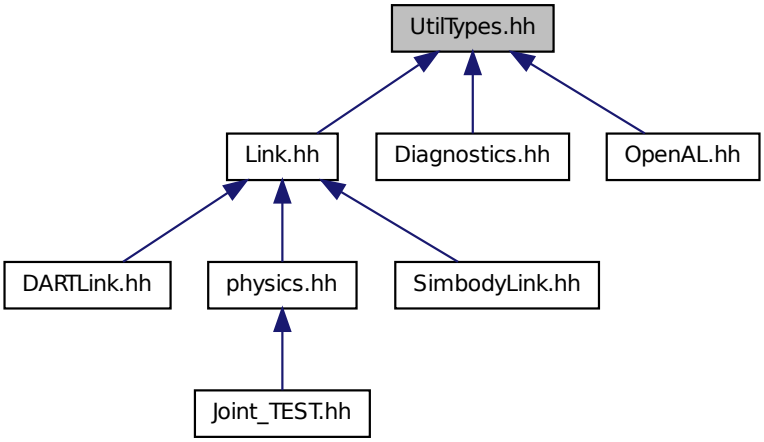
## 11.211 UtilTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for UtilTypes.hh:




This graph shows which files directly or indirectly include this file:




### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::util**

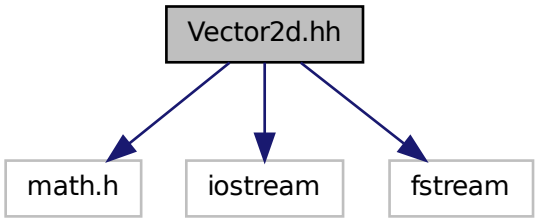
### Typedefs

- typedef boost::shared_ptr
  < DiagnosticTimer > **gazebo::util::DiagnosticTimerPtr**
- typedef boost::shared_ptr
  < OpenALSink > **gazebo::util::OpenALSinkPtr**
- typedef boost::shared_ptr
  < OpenALSource > **gazebo::util::OpenALSourcePtr**

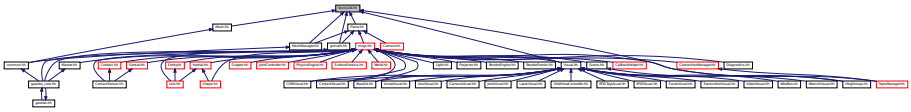
## 11.212 Vector2d.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
```

Include dependency graph for Vector2d.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::math::Vector2d**

  *Generic double x, y vector.*

### Namespaces

- namespace **gazebo**

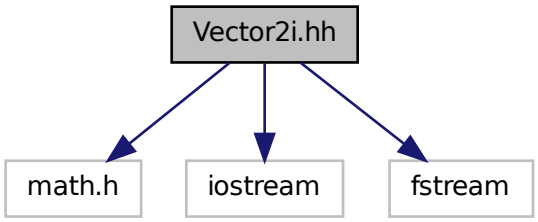
  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

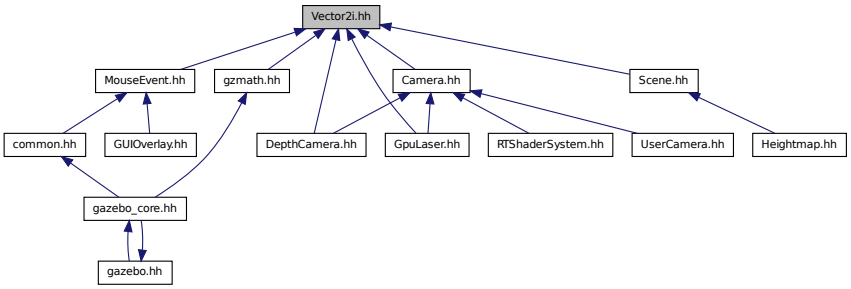
## 11.213 Vector2i.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
```

Include dependency graph for Vector2i.hh:




This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::math::Vector2i**

  *Generic integer x, y vector.*

### Namespaces

- namespace **gazebo**

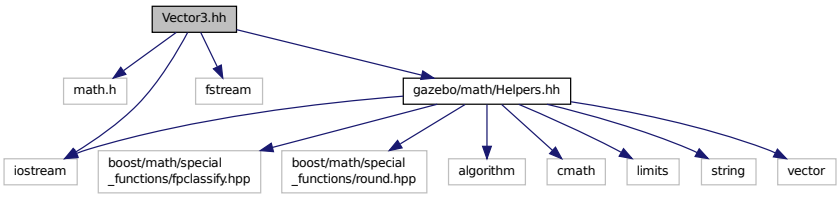
  *Forward declarations for the common classes.*
- namespace **gazebo::math**

  *Math namespace.*

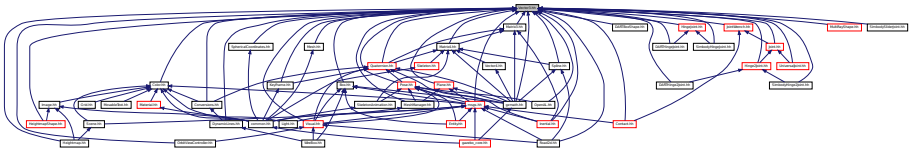
## 11.214 Vector3.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
#include "gazebo/math/Helpers.hh"
```

Include dependency graph for Vector3.hh:




This graph shows which files directly or indirectly include this file:



### Classes

- class **gazebo::math::Vector3**

  *The **Vector3** (p. 1091) class represents the generic vector containing 3 elements.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

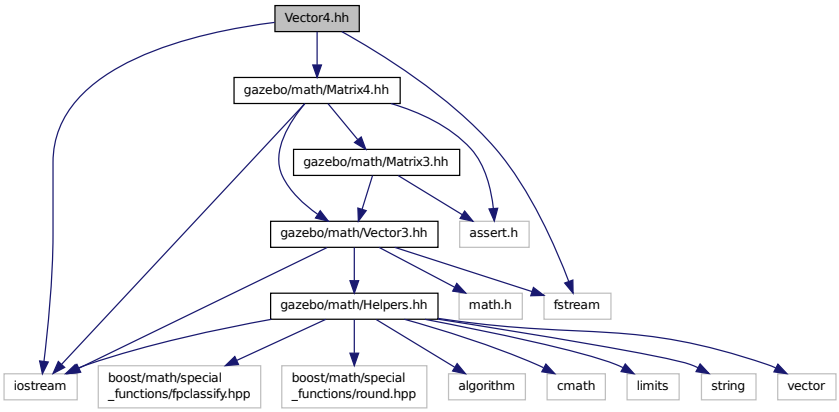
- namespace **gazebo::math**

  *Math namespace.*

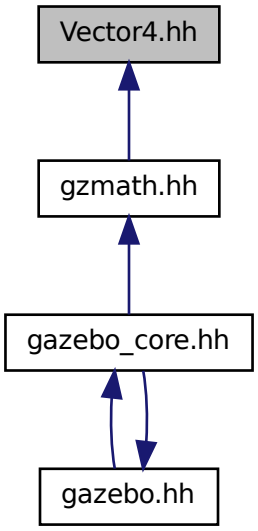
## 11.215 Vector4.hh File Reference

```
#include <iostream>
#include <fstream>
#include "gazebo/math/Matrix4.hh"
```

Include dependency graph for Vector4.hh:




This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::math::Vector4**

  *double Generic x, y, z, w vector*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

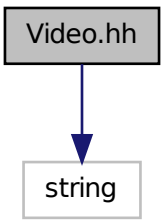
- namespace **gazebo::math**

  *Math namespace.*

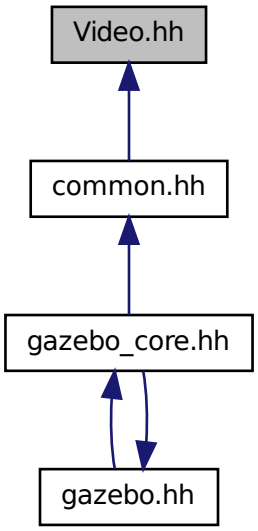
## 11.216 Video.hh File Reference

```
#include <string>
```

Include dependency graph for Video.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::common::Video**

  *Handle video encoding and decoding using libavcodec.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

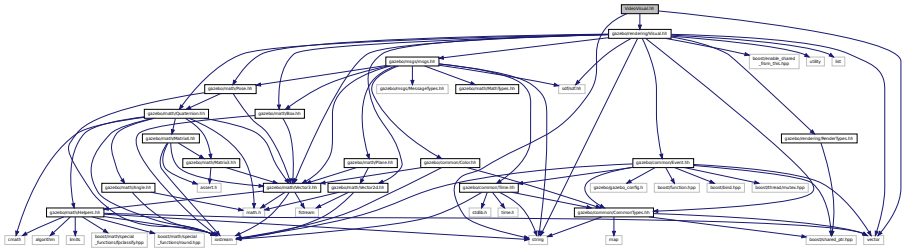
- namespace **gazebo::common**

  *Common namespace.*

## 11.217 VideoVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for VideoVisual.hh:



### Classes

- class **gazebo::rendering::VideoVisual**

  *A visual element that displays a video as a texture.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::common**

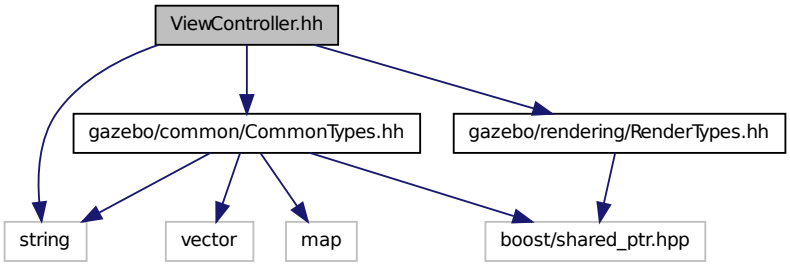
  *Common namespace.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

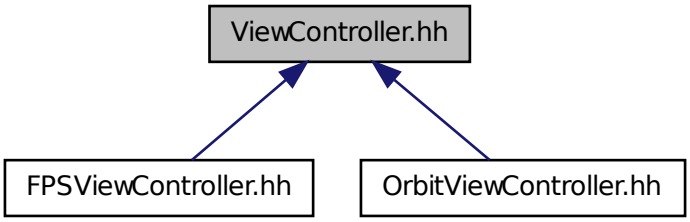
## 11.218 ViewController.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for ViewController.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::rendering::ViewController**

  *Base class for view controllers.*

### Namespaces

- namespace **gazebo**

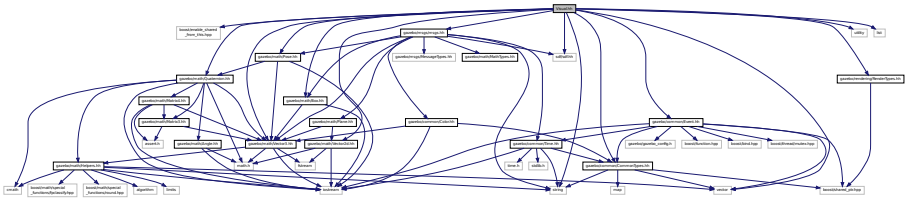
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

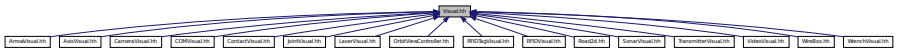
## 11.219 Visual.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Visual.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::rendering::Visual**

  *A renderable object.*

## Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

- namespace **gazebo::rendering**

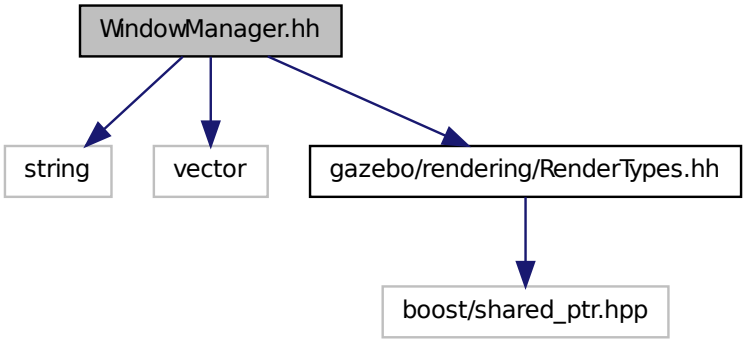
  *Rendering namespace.*

- namespace **Ogre**

# 11.220 WindowManager.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for WindowManager.hh:




### Classes

- class **gazebo::rendering::WindowManager**

  *Class to mangage render windows.*

### Namespaces

- namespace **gazebo**

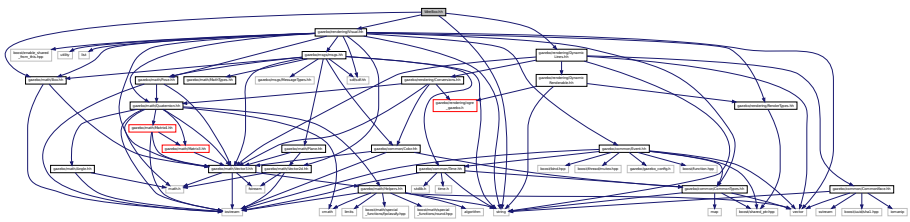
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*
- namespace **Ogre**

## 11.221 WireBox.hh File Reference

```
#include <string>
#include "gazebo/math/Box.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/DynamicLines.hh"
```

Include dependency graph for WireBox.hh:



### Classes

- class **gazebo::rendering::WireBox**

  *Draws a wireframe box.*

### Namespaces

- namespace **gazebo**

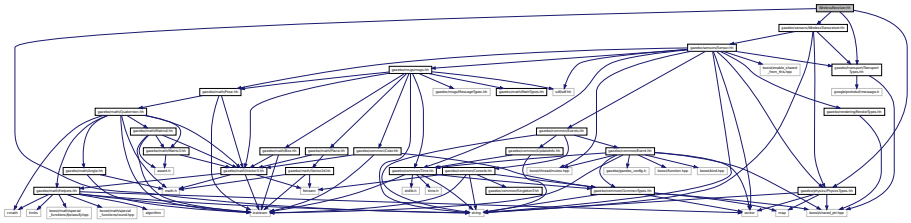
  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

## 11.222 WirelessReceiver.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/WirelessTransceiver.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for WirelessReceiver.hh:



### Classes

- class **gazebo::sensors::WirelessReceiver**

  ***Sensor*** (p. *837) class for receiving wireless signals.*

### Namespaces

- namespace **gazebo**

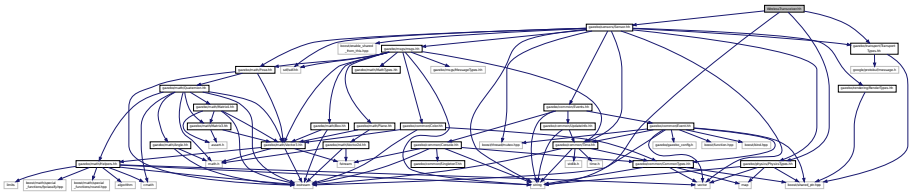
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

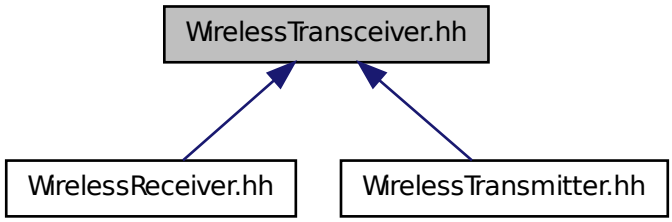
## 11.223 WirelessTransceiver.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for WirelessTransceiver.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::sensors::WirelessTransceiver**

  ***Sensor*** *(*p. *837) class for receiving wireless signals.*

### Namespaces

- namespace **gazebo**

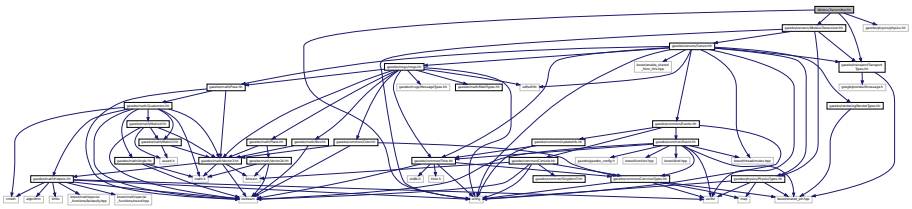
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

## 11.224 WirelessTransmitter.hh File Reference

```
#include <string>
#include "gazebo/physics/physics.hh"
#include "gazebo/sensors/WirelessTransceiver.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for WirelessTransmitter.hh:



### Classes

- class **gazebo::sensors::WirelessTransmitter**

  *Transmitter to send wireless signals.*

### Namespaces

- namespace **gazebo**

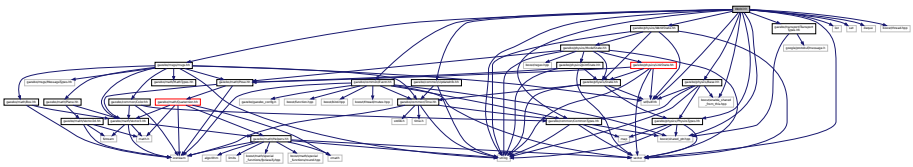
  *Forward declarations for the common classes.*
- namespace **gazebo::sensors**

  *Sensors namespace.*

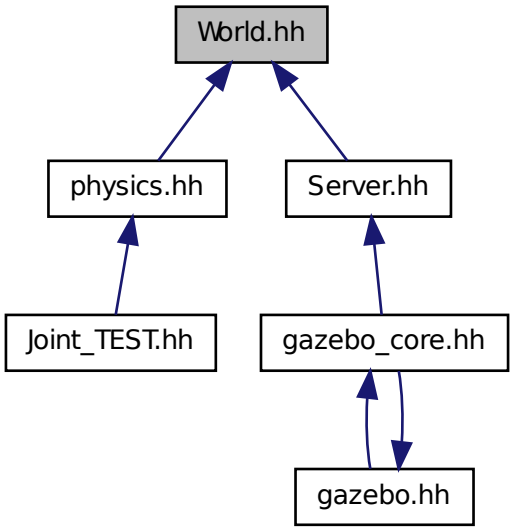
## 11.225 World.hh File Reference

```
#include <vector>
#include <list>
#include <set>
#include <deque>
#include <string>
#include <boost/thread.hpp>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include <sdf/sdf.hh>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/WorldState.hh"
```

Include dependency graph for World.hh:



This graph shows which files directly or indirectly include this file:




### Classes

- class **gazebo::physics::World**

  *The world provides access to all other object within a simulated environment.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*

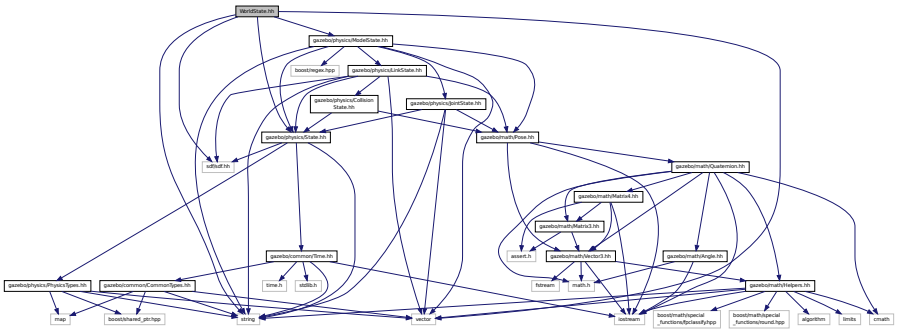
- namespace **gazebo::physics**

  *namespace for physics*

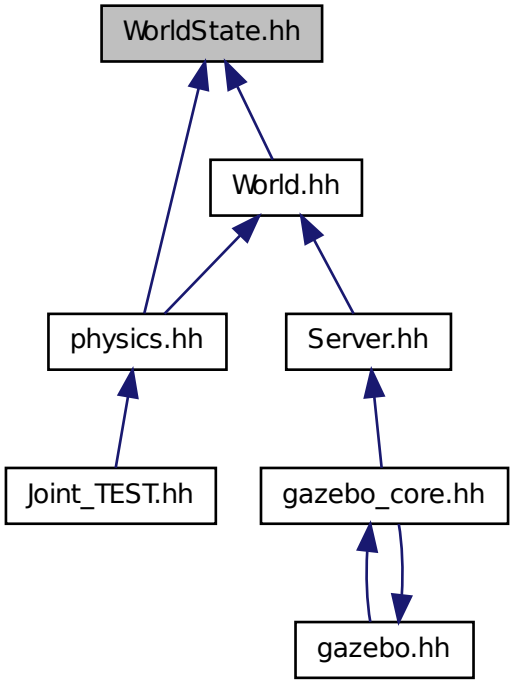
## 11.226 WorldState.hh File Reference

```
#include <string>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/physics/State.hh"
#include "gazebo/physics/ModelState.hh"
```

Include dependency graph for WorldState.hh:



This graph shows which files directly or indirectly include this file:




## Classes

- class **gazebo::physics::WorldState**

  *Store state information of a **physics::World*** (p. *1157*) *object.*

### Namespaces

- namespace **gazebo**

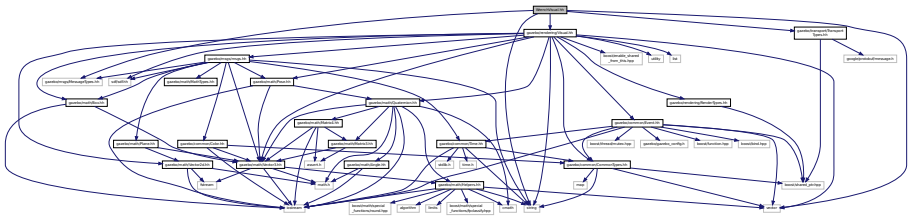
  *Forward declarations for the common classes.*
- namespace **gazebo::physics**

  *namespace for physics*

## 11.227 WrenchVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for WrenchVisual.hh:



### Classes

- class **gazebo::rendering::WrenchVisual**

  *Visualization for sonar data.*

### Namespaces

- namespace **gazebo**

  *Forward declarations for the common classes.*
- namespace **gazebo::rendering**

  *Rendering namespace.*

# Index


~Actor
gazebo::physics::Actor, 134
~Angle
gazebo::math::Angle, 139
~Animation
gazebo::common::Animation, 147
~ArrowVisual
gazebo::rendering::ArrowVisual, 151
~AssertionInternalError
gazebo::common::AssertionInternalError, 153
~AudioDecoder
gazebo::common::AudioDecoder, 153
~AxisVisual
gazebo::rendering::AxisVisual, 156
~BVHLoader
gazebo::common::BVHLoader, 180
~BallJoint
gazebo::physics::BallJoint, 158
~Base
gazebo::physics::Base, 164
~Box
gazebo::math::Box, 173
~BoxShape
gazebo::physics::BoxShape, 178
~COMVisual
gazebo::rendering::COMVisual, 246
~CallbackHelper
gazebo::transport::CallbackHelper, 182
~Camera
gazebo::rendering::Camera, 192
~CameraSensor
gazebo::sensors::CameraSensor, 214
~CameraVisual
gazebo::rendering::CameraVisual, 218
~ColladaLoader
gazebo::common::ColladaLoader, 219
~Collision
gazebo::physics::Collision, 222
~CollisionState
gazebo::physics::CollisionState, 231
~Color
gazebo::common::Color, 236
~Connection
gazebo::event::Connection, 247
gazebo::transport::Connection, 250
~Contact
gazebo::physics::Contact, 261
~ContactManager
gazebo::physics::ContactManager, 264
~ContactSensor
gazebo::sensors::ContactSensor, 269
~ContactVisual
gazebo::rendering::ContactVisual, 273
~CylinderShape
gazebo::physics::CylinderShape, 277
~DARTBallJoint
gazebo::physics::DARTBallJoint, 281
~DARTBoxShape
gazebo::physics::DARTBoxShape, 286
~DARTCollision
gazebo::physics::DARTCollision, 288
~DARTCylinderShape
gazebo::physics::DARTCylinderShape, 292
~DARTHeightmapShape
gazebo::physics::DARTHeightmapShape, 294
~DARTHinge2Joint
gazebo::physics::DARTHinge2Joint, 296
~DARTHingeJoint
gazebo::physics::DARTHingeJoint, 301
~DARTJoint
gazebo::physics::DARTJoint, 307
~DARTLink
gazebo::physics::DARTLink, 316
~DARTMeshShape
gazebo::physics::DARTMeshShape, 325
~DARTModel
gazebo::physics::DARTModel, 327
~DARTMultiRayShape
Bullet Physics, 76
~DARTPhysics
gazebo::physics::DARTPhysics, 332
~DARTPlaneShape
gazebo::physics::DARTPlaneShape, 337
~DARTRayShape
DART Physics, 74
~DARTScrewJoint
gazebo::physics::DARTScrewJoint, 341
~DARTSliderJoint
gazebo::physics::DARTSliderJoint, 347
~DARTSphereShape

gazebo::physics::DARTSphereShape, 351
~DARTUniversalJoint
gazebo::physics::DARTUniversalJoint, 354
~DepthCamera
gazebo::rendering::DepthCamera, 359
~DepthCameraSensor
gazebo::sensors::DepthCameraSensor, 363
~DiagnosticTimer
gazebo::util::DiagnosticTimer, 369
~DynamicLines
gazebo::rendering::DynamicLines, 372
~DynamicRenderable
gazebo::rendering::DynamicRenderable, 376
~Entity
gazebo::physics::Entity, 382
~Event
gazebo::event::Event, 392
~EventT
Events, 45
~Exception
gazebo::common::Exception, 417
~FPSViewController
gazebo::rendering::FPSViewController, 424
~ForceTorqueSensor
gazebo::sensors::ForceTorqueSensor, 420
~GUIOverlay
gazebo::rendering::GUIOverlay, 456
~GazeboGenerator
google::protobuf::compiler::cpp::GazeboGenerator, 426
~GpsSensor
gazebo::sensors::GpsSensor, 427
~GpuLaser
gazebo::rendering::GpuLaser, 432
~GpuRaySensor
gazebo::sensors::GpuRaySensor, 441
~Grid
gazebo::rendering::Grid, 451
~Gripper
gazebo::physics::Gripper, 454
~GzTerrainMatGen
gazebo::rendering::GzTerrainMatGen, 460
~Heightmap
gazebo::rendering::Heightmap, 462
~HeightmapShape
gazebo::physics::HeightmapShape, 467
~Hinge2Joint
gazebo::physics::Hinge2Joint, 472
~HingeJoint
gazebo::physics::HingeJoint, 473
~IOManager
gazebo::transport::IOManager, 495
~Image
gazebo::common::Image, 476
~ImuSensor
gazebo::sensors::ImuSensor, 481
~Inertial
gazebo::physics::Inertial, 486
~InternalError
gazebo::common::InternalError, 494
~Joint
gazebo::physics::Joint, 500
~JointState
gazebo::physics::JointState, 524
~JointVisual
gazebo::rendering::JointVisual, 529
~KeyFrame
gazebo::common::KeyFrame, 533
~LaserVisual
gazebo::rendering::LaserVisual, 535
~Light
gazebo::rendering::Light, 537
~Link
gazebo::physics::Link, 547
~LinkState
gazebo::physics::LinkState, 565
~MapShape
gazebo::physics::MapShape, 581
~Master
gazebo::Master, 584
~Material
gazebo::common::Material, 588
~Matrix3
gazebo::math::Matrix3, 596
~Matrix4
gazebo::math::Matrix4, 601
~Mesh
gazebo::common::Mesh, 608
~MeshCSG
gazebo::common::MeshCSG, 614
~MeshLoader
gazebo::common::MeshLoader, 615
~MeshShape
gazebo::physics::MeshShape, 622
~Model
gazebo::physics::Model, 628
~ModelPlugin
gazebo::ModelPlugin, 640
~ModelState
gazebo::physics::ModelState, 644
~MovableText
gazebo::rendering::MovableText, 655
~MultiCameraSensor
gazebo::sensors::MultiCameraSensor, 661
~MultiRayShape
gazebo::physics::MultiRayShape, 667
~Node
gazebo::transport::Node, 674

~NodeAnimation
gazebo::common::NodeAnimation, 681
~NodeTransform
gazebo::common::NodeTransform, 686
~Noise
gazebo::sensors::Noise, 690
~NumericAnimation
gazebo::common::NumericAnimation, 693
~NumericKeyFrame
gazebo::common::NumericKeyFrame, 694
~OpenALSink
gazebo::util::OpenALSink, 698
~OpenALSource
gazebo::util::OpenALSource, 699
~OrbitViewController
gazebo::rendering::OrbitViewController, 704
~PID
gazebo::common::PID, 723
~PhysicsEngine
gazebo::physics::PhysicsEngine, 710
~Plane
gazebo::math::Plane, 727
~PlaneShape
gazebo::physics::PlaneShape, 730
~PluginT
gazebo::PluginT, 733
~Pose
gazebo::math::Pose, 737
~PoseAnimation
gazebo::common::PoseAnimation, 745
~PoseKeyFrame
gazebo::common::PoseKeyFrame, 747
~Projector
gazebo::rendering::Projector, 749
~Publication
gazebo::transport::Publication, 751
~PublicationTransport
gazebo::transport::PublicationTransport, 756
~Publisher
gazebo::transport::Publisher, 758
~Quaternion
gazebo::math::Quaternion, 765
~RFIDSensor
gazebo::sensors::RFIDSensor, 797
~RFIDTag
gazebo::sensors::RFIDTag, 800
~RFIDTagVisual
gazebo::rendering::RFIDTagVisual, 802
~RFIDVisual
gazebo::rendering::RFIDVisual, 803
~RaySensor
gazebo::sensors::RaySensor, 781
~RayShape
gazebo::physics::RayShape, 788
~Road
gazebo::physics::Road, 805
~Road2d
gazebo::rendering::Road2d, 806
~RotationSpline
gazebo::math::RotationSpline, 807
~SM2Profile
gazebo::rendering::GzTerrainMatGen::SM2Profile, 977
~STLLoader
gazebo::common::STLLoader, 1003
~Scene
gazebo::rendering::Scene, 819
~ScrewJoint
gazebo::physics::ScrewJoint, 833
~SelectionObj
Rendering, 83
~Sensor
gazebo::sensors::Sensor, 840
~SensorPlugin
gazebo::SensorPlugin, 854
~Server
gazebo::Server, 856
~Shape
gazebo::physics::Shape, 863
~SimbodyBallJoint
gazebo::physics::SimbodyBallJoint, 867
~SimbodyBoxShape
gazebo::physics::SimbodyBoxShape, 872
~SimbodyCollision
gazebo::physics::SimbodyCollision, 874
~SimbodyCylinderShape
gazebo::physics::SimbodyCylinderShape, 877
~SimbodyHeightmapShape
gazebo::physics::SimbodyHeightmapShape, 879
~SimbodyHinge2Joint
gazebo::physics::SimbodyHinge2Joint, 881
~SimbodyHingeJoint
gazebo::physics::SimbodyHingeJoint, 887
~SimbodyJoint
gazebo::physics::SimbodyJoint, 894
~SimbodyLink
gazebo::physics::SimbodyLink, 903
~SimbodyMeshShape
gazebo::physics::SimbodyMeshShape, 911
~SimbodyModel
gazebo::physics::SimbodyModel, 913
~SimbodyMultiRayShape
gazebo::physics::SimbodyMultiRayShape, 915
~SimbodyPhysics
gazebo::physics::SimbodyPhysics, 918
~SimbodyPlaneShape
gazebo::physics::SimbodyPlaneShape, 926
~SimbodyRayShape

gazebo::physics::SimbodyRayShape, 928
~SimbodyScrewJoint
gazebo::physics::SimbodyScrewJoint, 932
~SimbodySliderJoint
gazebo::physics::SimbodySliderJoint, 938
~SimbodySphereShape
gazebo::physics::SimbodySphereShape, 944
~SimbodyUniversalJoint
gazebo::physics::SimbodyUniversalJoint, 947
~SingletonT
SingletonT, 953
~Skeleton
gazebo::common::Skeleton, 955
~SkeletonAnimation
gazebo::common::SkeletonAnimation, 961
~SkeletonNode
gazebo::common::SkeletonNode, 967
~SliderJoint
gazebo::physics::SliderJoint, 974
~SonarSensor
gazebo::sensors::SonarSensor, 979
~SonarVisual
gazebo::rendering::SonarVisual, 983
~SpawnJointOptions
Joint_TEST::SpawnJointOptions, 984
~SphereShape
gazebo::physics::SphereShape, 987
~SphericalCoordinates
gazebo::common::SphericalCoordinates, 990
~Spline
gazebo::math::Spline, 994
~State
gazebo::physics::State, 999
~SubMesh
gazebo::common::SubMesh, 1007
~Subscriber
gazebo::transport::Subscriber, 1017
~SubscriptionTransport
gazebo::transport::SubscriptionTransport, 1019
~SurfaceParams
gazebo::physics::SurfaceParams, 1021
~SystemPlugin
gazebo::SystemPlugin, 1031
~Time
gazebo::common::Time, 1036
~Timer
gazebo::common::Timer, 1053
~TransmitterVisual
gazebo::rendering::TransmitterVisual, 1063
~UniversalJoint
gazebo::physics::UniversalJoint, 1064
~UserCamera
gazebo::rendering::UserCamera, 1068
~Vector2d
gazebo::math::Vector2d, 1076
~Vector2i
gazebo::math::Vector2i, 1085
~Vector3
gazebo::math::Vector3, 1095
~Vector4
gazebo::math::Vector4, 1108
~Video
gazebo::common::Video, 1115
~VideoVisual
gazebo::rendering::VideoVisual, 1117
~ViewController
gazebo::rendering::ViewController, 1119
~Visual
gazebo::rendering::Visual, 1127
~WindowManager
gazebo::rendering::WindowManager, 1144
~WireBox
gazebo::rendering::WireBox, 1147
~WirelessReceiver
gazebo::sensors::WirelessReceiver, 1149
~WirelessTransceiver
gazebo::sensors::WirelessTransceiver, 1151
~WirelessTransmitter
gazebo::sensors::WirelessTransmitter, 1155
~World
gazebo::physics::World, 1160
~WorldPlugin
gazebo::WorldPlugin, 1170
~WorldState
gazebo::physics::WorldState, 1173
~WrenchVisual
gazebo::rendering::WrenchVisual, 1178
_setupGeometry
gazebo::rendering::MovableText, 655
_updateColors
gazebo::rendering::MovableText, 655

a
gazebo::common::Color, 244
ABGR
gazebo::common::Color, 236
ACTOR
gazebo::physics::Base, 163
ADD
gazebo::common::Material, 587
ARGB
gazebo::common::Color, 236
AcceptCallback
gazebo::transport::Connection, 250
active
gazebo::physics::Actor, 135
gazebo::sensors::Sensor, 847
Actor

gazebo::physics::Actor, 133
Actor.hh, 1179
Actor_V
gazebo::physics, 116
ActorPtr
gazebo::physics, 116
Add
gazebo::util::LogRecord, 575
add_plugin
gazebo, 101
add_search_path_suffix
Common, 39
AddAnimation
gazebo::common::Skeleton, 955
AddCallback
gazebo::transport::PublicationTransport, 756
AddChild
gazebo::common::SkeletonNode, 967
gazebo::physics::Base, 164
AddChildJoint
gazebo::physics::Link, 547
AddContact
gazebo::physics::Collision, 223
AddDARTChildJoint
gazebo::physics::DARTLink, 316
addEntity
gazebo::event::Events, 403
AddForce
gazebo::physics::DARTLink, 316
gazebo::physics::Link, 547
gazebo::physics::SimbodyLink, 903
AddForceAtRelativePosition
gazebo::physics::DARTLink, 317
gazebo::physics::Link, 547
gazebo::physics::SimbodyLink, 903
AddForceAtWorldPosition
gazebo::physics::DARTLink, 317
gazebo::physics::Link, 548
gazebo::physics::SimbodyLink, 904
AddGazeboPaths
gazebo::common::SystemPaths, 1026
AddIndex
gazebo::common::SubMesh, 1007
AddJoint
gazebo::physics::JointController, 521
AddKeyFrame
gazebo::common::NodeAnimation, 681
gazebo::common::SkeletonAnimation, 961
AddMaterial
gazebo::common::Mesh, 608
AddMesh
gazebo::common::MeshManager, 617
AddModelPaths
gazebo::common::SystemPaths, 1026
AddNode
gazebo::transport::TopicManager, 1056
AddNodeAssignment
gazebo::common::SubMesh, 1007
AddNodeToProcess
gazebo::transport::TopicManager, 1057
AddNormal
gazebo::common::SubMesh, 1007
AddOgrePaths
gazebo::common::SystemPaths, 1027
AddParentJoint
gazebo::physics::Link, 548
AddPluginPaths
gazebo::common::SystemPaths, 1027
AddPoint
gazebo::math::RotationSpline, 808
gazebo::math::Spline, 995
gazebo::rendering::DynamicLines, 373
AddPublisher
gazebo::transport::Publication, 752
AddRawTransform
gazebo::common::SkeletonNode, 967
AddRay
Bullet Physics, 76
gazebo::physics::MultiRayShape, 667
gazebo::physics::SimbodyMultiRayShape, 915
AddRelativeForce
gazebo::physics::DARTLink, 317
gazebo::physics::Link, 548
gazebo::physics::SimbodyLink, 904
AddRelativeTorque
gazebo::physics::DARTLink, 317
gazebo::physics::Link, 548
gazebo::physics::SimbodyLink, 904
AddResourcePath
gazebo::rendering::RenderEngine, 793
AddScene
gazebo::rendering::RTShaderSystem, 812
AddSearchPathSuffix
gazebo::common::SystemPaths, 1027
AddSubMesh
gazebo::common::Mesh, 608
AddSubscription
gazebo::transport::Publication, 752
AddTag
gazebo::sensors::RFIDSensor, 797
addTechnique
gazebo::rendering::GzTerrainMatGen::SM2Profile, 977
AddTexCoord
gazebo::common::SubMesh, 1007
AddTime
gazebo::common::Animation, 147
AddTorque

gazebo::physics::DARTLink, 318
gazebo::physics::Link, 548
gazebo::physics::SimbodyLink, 904
AddTransport
gazebo::transport::Publication, 752
AddType
gazebo::physics::Base, 164
AddVertNodeWeight
gazebo::common::Skeleton, 955
AddVertex
gazebo::common::SubMesh, 1008
AddVisual
gazebo::rendering::Scene, 819
Advertise
gazebo::transport::ConnectionManager, 256
gazebo::transport::Node, 674
gazebo::transport::TopicManager, 1057
alt
gazebo::common::MouseEvent, 651
ambient
gazebo::common::Material, 593
anchorLink
gazebo::physics::Joint, 515
anchorPos
gazebo::physics::Joint, 515
anchorPose
gazebo::physics::Joint, 515
Angle
gazebo::math::Angle, 139
Angle.hh, 1180
GZ_DTOR, 1182
GZ_NORMALIZE, 1182
GZ_RTOD, 1182
angularAccel
gazebo::physics::Link, 562
animState
gazebo::rendering::Camera, 210
Animation
gazebo::common::Animation, 147
animation
gazebo::physics::Entity, 389
Animation.hh, 1183
AnimationComplete
gazebo::rendering::Camera, 193
gazebo::rendering::UserCamera, 1068
animationConnection
gazebo::physics::Entity, 389
AnimationPtr
gazebo::common, 104
animationStartPose
gazebo::physics::Entity, 389
animations
gazebo::common::SkeletonAnimation, 963
anims
gazebo::common::Skeleton, 959
Apply
gazebo::sensors::Noise, 690
ApplyDamping
gazebo::physics::DARTJoint, 307
gazebo::physics::Joint, 501
applyDamping
gazebo::physics::Joint, 515
ApplyShadows
gazebo::rendering::RTShaderSystem, 812
AreConnected
gazebo::physics::DARTJoint, 307
gazebo::physics::Joint, 501
gazebo::physics::SimbodyJoint, 894
ArrowVisual
gazebo::rendering::ArrowVisual, 150
ArrowVisual.hh, 1184
ArrowVisualPtr
gazebo::rendering, 121
Assert.hh, 1185
GZ_ASSERT, 1186
AssertionInternalError
gazebo::common::AssertionInternalError, 152
AsyncRead
gazebo::transport::Connection, 250
Attach
gazebo::physics::DARTJoint, 307
gazebo::physics::Joint, 501
Rendering, 83
AttachAxes
gazebo::rendering::Visual, 1127
AttachCameraToImage
gazebo::rendering::GUIOverlay, 456, 457
AttachEntity
gazebo::rendering::RTShaderSystem, 812
AttachLineVertex
gazebo::rendering::Visual, 1127
AttachMesh
gazebo::rendering::Visual, 1128
AttachObject
gazebo::rendering::Visual, 1128
AttachStaticModel
gazebo::physics::Link, 549
gazebo::physics::Model, 628
AttachToVisual
gazebo::rendering::Camera, 193
AttachToVisualImpl
gazebo::rendering::Camera, 193, 194
gazebo::rendering::UserCamera, 1068
AttachViewport
gazebo::rendering::RTShaderSystem, 812
AttachVisual
gazebo::rendering::Visual, 1128
attachedModels

gazebo::physics::Model, 638
attachedModelsOffset
gazebo::physics::Link, 562
gazebo::physics::Model, 638
Attribute
gazebo::physics::Joint, 500
AudioDecoder
gazebo::common::AudioDecoder, 153
AudioDecoder.hh, 1187
autoCalc
gazebo::math::RotationSpline, 810
gazebo::math::Spline, 997
autoStart
gazebo::physics::Actor, 135
axis
Joint_TEST::SpawnJointOptions, 984
AxisVisual
gazebo::rendering::AxisVisual, 156
AxisVisual.hh, 1188
AxisVisualPtr
gazebo::rendering, 121

b
gazebo::common::Color, 244
BALL_JOINT
gazebo::physics::Base, 163
BASE
gazebo::physics::Base, 163
BAYER_GBRG8
gazebo::common::Image, 476
BAYER_GRBG8
gazebo::common::Image, 476
BAYER_RGGB8
gazebo::common::Image, 476
BAYER_RGGR8
gazebo::common::Image, 476
BGR_INT16
gazebo::common::Image, 476
BGR_INT32
gazebo::common::Image, 476
BGR_INT8
gazebo::common::Image, 476
BGRA
gazebo::common::Color, 236
BGRA_INT8
gazebo::common::Image, 476
BLEND_COUNT
gazebo::common::Material, 587
BLINN
gazebo::common::Material, 588
BOX_SHAPE
gazebo::physics::Base, 163
BVHLoader
gazebo::common::BVHLoader, 180
BVHLoader.hh, 1194
X_POSITION, 1196
X_ROTATION, 1196
Y_POSITION, 1196
Y_ROTATION, 1196
Z_POSITION, 1196
Z_ROTATION, 1196
BackupState
gazebo::physics::DARTModel, 327
BallJoint
gazebo::physics::BallJoint, 158
BallJoint.hh, 1188
Base
gazebo::physics::Base, 164
Base.hh, 1189
Base64.hh, 1190
Base64Decode, 1192
Base64Encode, 1192
Base64Decode
Base64.hh, 1192
Base64Encode
Base64.hh, 1192
Base_V
gazebo::physics, 116
BasePtr
gazebo::physics, 116
bayerFrameBuffer
gazebo::rendering::Camera, 210
bindShapeTransform
gazebo::common::Skeleton, 959
Black
gazebo::common::Color, 244
BlendMode
gazebo::common::Material, 587
blendMode
gazebo::common::Material, 593
BlendModeStr
gazebo::common::Material, 593
Blue
gazebo::common::Color, 244
body1Force
gazebo::physics::JointWrench, 530
body1Torque
gazebo::physics::JointWrench, 530
body2Force
gazebo::physics::JointWrench, 531
body2Torque
gazebo::physics::JointWrench, 531
bonePosePub
gazebo::physics::Actor, 135
BooleanOperation
gazebo::common::MeshCSG, 613
boost, 99
bounce

gazebo::physics::SurfaceParams, 1022
bounceThreshold
gazebo::physics::SurfaceParams, 1022
Box
gazebo::math::Box, 173
Box.hh, 1192
BoxShape
gazebo::physics::BoxShape, 178
BoxShape.hh, 1193
BoxShapePtr
gazebo::physics, 116
build
gazebo::common::Animation, 149
BuildInterpolationSplines
gazebo::common::PoseAnimation, 745
BuildNodeMap
gazebo::common::Skeleton, 956
Bullet Physics, 76
∼DARTMultiRayShape, 76
AddRay, 76
DARTMultiRayShape, 76
UpdateRays, 77
button
gazebo::common::MouseEvent, 651
ButtonCallback
gazebo::rendering::GUIOverlay, 457
Buttons
gazebo::common::MouseEvent, 651
buttons
gazebo::common::MouseEvent, 651

CATEGORY_COUNT
gazebo::sensors, 126
CFM
gazebo::physics::Joint, 500
COLLISION
gazebo::physics::Base, 163
COMVisual
gazebo::rendering::COMVisual, 245
COMVisual.hh, 1208
COMVisualPtr
gazebo::rendering, 121
COR3_MAX
STLLoader.hh, 1379
CYLINDER_SHAPE
gazebo::physics::Base, 163
CacheForceTorque
gazebo::physics::Joint, 501
gazebo::physics::SimbodyJoint, 894
CallbackHelper
gazebo::transport::CallbackHelper, 182
CallbackHelper.hh, 1196
CallbackHelperPtr
Transport, 94
CallbackHelperT
gazebo::transport::CallbackHelperT, 184
Camera
gazebo::rendering::Camera, 192
camera
gazebo::rendering::Camera, 210
gazebo::rendering::ViewController, 1121
Camera.hh, 1198
cameraCount
gazebo::rendering::GpuLaser, 437
cameraElem
gazebo::sensors::GpuRaySensor, 449
CameraPtr
gazebo::rendering, 121
CameraSensor
gazebo::sensors::CameraSensor, 214
CameraSensor.hh, 1199
CameraSensor_V
gazebo::sensors, 125
CameraSensorPtr
gazebo::sensors, 125
CameraVisual
gazebo::rendering::CameraVisual, 218
CameraVisual.hh, 1199
CameraVisualPtr
gazebo::rendering, 121
Cancel
gazebo::transport::Connection, 250
captureData
gazebo::rendering::Camera, 210
captureDataOnce
gazebo::rendering::Camera, 210
cegui.h, 1200
Center
gazebo::common::Mesh, 608
gazebo::common::SubMesh, 1008
cfm
gazebo::physics::SurfaceParams, 1022
cgVisuals
gazebo::physics::Link, 562
CheckAndTruncateForce
gazebo::physics::Joint, 501
chfov
gazebo::rendering::GpuLaser, 437
childLink
gazebo::physics::Joint, 515
childLinkPose
Joint_TEST::SpawnJointOptions, 984
children
gazebo::common::SkeletonNode, 972
gazebo::physics::Base, 171
childrenEnd
gazebo::physics::Base, 171
clamp

Math, 50
Classes for physics and dynamics, 65
create_world, 69
EntityTypename, 72
fini, 69
GZ_REGISTER_PHYSICS_ENGINE, 69
get_world, 69
getUniqueId, 70
init_world, 70
init_worlds, 70
load, 70
load_world, 70
load_worlds, 70
pause_world, 71
pause_worlds, 71
PhysicsFactoryFn, 69
remove_worlds, 71
run_world, 71
run_worlds, 71
stop_world, 71
stop_worlds, 72
worlds_running, 72
Clear
gazebo::math::RotationSpline, 808
gazebo::math::Spline, 995
gazebo::physics::ContactManager, 264
gazebo::physics::World, 1160
gazebo::rendering::DynamicLines, 373
gazebo::rendering::RTShaderSystem, 813
gazebo::rendering::Scene, 819
clear_buffers
Transport, 94
ClearBuffers
gazebo::transport::TopicManager, 1057
ClearGazeboPaths
gazebo::common::SystemPaths, 1027
ClearModelPaths
gazebo::common::SystemPaths, 1027
ClearModels
gazebo::physics::World, 1160
ClearOgrePaths
gazebo::common::SystemPaths, 1027
ClearParent
gazebo::rendering::Visual, 1128
ClearPluginPaths
gazebo::common::SystemPaths, 1027
Clone
gazebo::rendering::Visual, 1128
CloneVisual
gazebo::rendering::Scene, 819
coeffs
gazebo::math::Spline, 997
ColladaLoader
gazebo::common::ColladaLoader, 219
ColladaLoader.hh, 1201
collideWithoutContact
gazebo::physics::SurfaceParams, 1022
collideWithoutContactBitmask
gazebo::physics::SurfaceParams, 1022
Collision
gazebo::physics::Collision, 222
Collision.hh, 1202
collision1
gazebo::physics::Contact, 262
collision2
gazebo::physics::Contact, 262
Collision_V
gazebo::physics, 116
collisionNames
gazebo::physics::ContactPublisher, 267
collisionParent
gazebo::physics::Shape, 864
CollisionPtr
gazebo::physics, 116
CollisionState
gazebo::physics::CollisionState, 231
CollisionState.hh, 1203
collisions
gazebo::physics::ContactPublisher, 267
Color
gazebo::common::Color, 236
Color.hh, 1203
ColorErr
Common, 39
ColorMsg
Common, 39
Common, 33
add_search_path_suffix, 39
ColorErr, 39
ColorMsg, 39
DownloadDependencies, 39
find_file, 40
find_file_path, 40
Fini, 40
get_sha1, 40
GetDBConfig, 41
GetModelConfig, 41
GetModelFile, 41
GetModelName, 41
GetModelPath, 42
GetModels, 42
GetQuiet, 42
GetURI, 42
gzclr_end, 37
gzclr_start, 37
gzdbg, 37
gzerr, 37
gzlog, 37

gzmsg, 38
gzthrow, 38
gzwarn, 38
HasModel, 43
Init, 43
IsInitialized, 43
load, 43
Log, 43
MODEL_PLUGIN, 38
NullStream, 39
PixelFormatNames, 44
PluginType, 38
SENSOR_PLUGIN, 38
SYSTEM_PLUGIN, 38
SetQuiet, 43
Start, 44
VISUAL_PLUGIN, 38
WORLD_PLUGIN, 38
CommonIface.hh, 1204
CommonTypes.hh, 1206
GAZEBO_DEPRECATED, 1208
GAZEBO_FORCEINLINE, 1208
NULL, 1208
ComputeScopedName
gazebo::physics::Base, 164
Connect
Events, 45
gazebo::transport::Connection, 250
ConnectAddEntity
gazebo::event::Events, 395
ConnectCreateEntity
gazebo::event::Events, 396
ConnectCreateScene
gazebo::rendering::Events, 406
ConnectDeleteEntity
gazebo::event::Events, 396
ConnectDiagTimerStart
gazebo::event::Events, 396
ConnectDiagTimerStop
gazebo::event::Events, 397
ConnectEnabled
gazebo::physics::Link, 549
ConnectJointUpdate
gazebo::physics::Joint, 501
ConnectNewDepthFrame
gazebo::rendering::DepthCamera, 359
ConnectNewImageFrame
gazebo::rendering::Camera, 194
ConnectNewLaserFrame
gazebo::rendering::GpuLaser, 432
gazebo::sensors::GpuRaySensor, 442
ConnectNewLaserScans
gazebo::physics::MultiRayShape, 667
ConnectNewRGBPointCloud
gazebo::rendering::DepthCamera, 360
ConnectPause
gazebo::event::Events, 397
ConnectPostRender
gazebo::event::Events, 397
ConnectPreRender
gazebo::event::Events, 397
ConnectPubToSub
gazebo::transport::TopicManager, 1057
ConnectRemoveScene
gazebo::rendering::Events, 406
ConnectRender
gazebo::event::Events, 398
ConnectSetSelectedEntity
gazebo::event::Events, 398
ConnectSigInt
gazebo::event::Events, 398
ConnectStep
gazebo::event::Events, 399
ConnectStop
gazebo::event::Events, 399
ConnectSubToPub
gazebo::transport::TopicManager, 1057
ConnectSubscribers
gazebo::transport::TopicManager, 1057
ConnectToRemoteHost
gazebo::transport::ConnectionManager, 256
ConnectToShutdown
gazebo::transport::Connection, 250
ConnectUpdate
gazebo::sensors::ForceTorqueSensor, 420
gazebo::sensors::SonarSensor, 979
ConnectUpdated
gazebo::sensors::Sensor, 841
ConnectWorldCreated
gazebo::event::Events, 399
ConnectWorldUpdateBegin
gazebo::event::Events, 399
ConnectWorldUpdateEnd
gazebo::event::Events, 400
Connection
gazebo::event::Connection, 247
gazebo::transport::Connection, 250
Connection.hh, 1209
HEADER_LENGTH, 1211
Connection_V
gazebo::event, 105
ConnectionCount
Events, 46
ConnectionManager.hh, 1211
ConnectionPtr
gazebo::event, 105
gazebo::transport, 129
connections

gazebo::physics::Entity, 389
gazebo::rendering::Camera, 211
gazebo::sensors::Sensor, 847
Console.hh, 1213
constraint
gazebo::physics::SimbodyJoint, 899
Contact
gazebo::physics::Contact, 261
contact
gazebo::physics::SimbodyPhysics, 924
Contact.hh, 1214
MAX_COLLIDE_RETURNS, 1215
MAX_CONTACT_JOINTS, 1215
contactFiducial
gazebo::physics::RayShape, 791
contactLen
gazebo::physics::RayShape, 791
ContactManager
gazebo::physics::ContactManager, 264
contactManager
gazebo::physics::PhysicsEngine, 719
ContactManager.hh, 1216
ContactPtr
gazebo::physics, 116
contactRetro
gazebo::physics::RayShape, 791
ContactSensor
gazebo::sensors::ContactSensor, 269
ContactSensor.hh, 1217
ContactSensor_V
gazebo::sensors, 125
ContactSensorPtr
gazebo::sensors, 125
ContactVisual
gazebo::rendering::ContactVisual, 273
ContactVisual.hh, 1217
ContactVisualPtr
gazebo::rendering, 121
contacts
gazebo::physics::ContactPublisher, 267
control
gazebo::common::MouseEvent, 651
ConvPose
DART Physics, 74
ConvQuat
DART Physics, 74, 75
ConvVec3
DART Physics, 75
Conversions.hh, 1218
Convert
gazebo::common::SphericalCoordinates, 991
gazebo::rendering::Conversions, 274, 275
Messages, 56–59
ConvertPixelFormat
gazebo::common::Image, 476
CoordPoseSolve
gazebo::math::Pose, 737
CoordPositionAdd
gazebo::math::Pose, 737, 738
CoordPositionSub
gazebo::math::Pose, 738
CoordRotationAdd
gazebo::math::Pose, 738
CoordRotationSub
gazebo::math::Pose, 738
CopyNormals
gazebo::common::SubMesh, 1008
CopyVertices
gazebo::common::SubMesh, 1008
Correct
gazebo::math::Pose, 739
gazebo::math::Quaternion, 765
gazebo::math::Vector3, 1095
count
gazebo::physics::Contact, 262
Create
gazebo::PluginT, 733
create_scene
Rendering, 84
create_sensor
Sensors, 89
create_world
Classes for physics and dynamics, 69
CreateBoolean
gazebo::common::MeshCSG, 614
CreateBox
gazebo::common::MeshManager, 617
CreateCamera
gazebo::common::MeshManager, 617
gazebo::rendering::Scene, 819
CreateCollision
gazebo::physics::DARTPhysics, 332
gazebo::physics::PhysicsEngine, 710
gazebo::physics::SimbodyPhysics, 918
CreateCone
gazebo::common::MeshManager, 618
CreateCylinder
gazebo::common::MeshManager, 618
CreateDepthCamera
gazebo::rendering::Scene, 819
CreateDepthTexture
gazebo::rendering::DepthCamera, 360
CreateDynamicLine
gazebo::rendering::Visual, 1128
CreateFilter
gazebo::physics::ContactManager, 264, 265
CreateGpuLaser
gazebo::rendering::Scene, 820

CreateGrid
gazebo::rendering::Scene, 820
CreateJoint
gazebo::physics::DARTPhysics, 332
gazebo::physics::PhysicsEngine, 711
gazebo::physics::SimbodyPhysics, 918
CreateKeyFrame
gazebo::common::NumericAnimation, 693
gazebo::common::PoseAnimation, 745
CreateLaserTexture
gazebo::rendering::GpuLaser, 432
CreateLink
gazebo::physics::DARTPhysics, 332
gazebo::physics::PhysicsEngine, 711
gazebo::physics::SimbodyPhysics, 918
CreateModel
gazebo::physics::DARTPhysics, 333
gazebo::physics::PhysicsEngine, 711
gazebo::physics::SimbodyPhysics, 919
CreatePlane
gazebo::common::MeshManager, 618
gazebo::physics::DARTPlaneShape, 337
gazebo::physics::PlaneShape, 730
gazebo::physics::SimbodyPlaneShape, 926
CreateRenderTexture
gazebo::rendering::Camera, 195
CreateRequest
Messages, 59
CreateScene
gazebo::rendering::RenderEngine, 793
createScene
gazebo::rendering::Events, 407
CreateSensor
gazebo::sensors::SensorManager, 851
CreateShape
gazebo::physics::DARTPhysics, 333
gazebo::physics::PhysicsEngine, 711
gazebo::physics::SimbodyPhysics, 919
CreateSink
gazebo::util::OpenAL, 696
CreateSource
gazebo::util::OpenAL, 696
CreateSphere
gazebo::common::MeshManager, 619
CreateTube
gazebo::common::MeshManager, 619
CreateUserCamera
gazebo::rendering::Scene, 820
CreateVertexDeclaration
gazebo::rendering::DynamicRenderable, 376
CreateWindow
gazebo::rendering::GUIOverlay, 457
gazebo::rendering::WindowManager, 1145
Cross
gazebo::math::Vector2d, 1077
gazebo::math::Vector2i, 1085
gazebo::math::Vector3, 1095
cvfov
gazebo::rendering::GpuLaser, 437
CylinderShape
gazebo::physics::CylinderShape, 277
CylinderShape.hh, 1219
CylinderShapePtr
gazebo::physics, 116

d
gazebo::math::Plane, 728
DART Physics, 73
∼DARTRayShape, 74
ConvPose, 74
ConvQuat, 74, 75
ConvVec3, 75
DARTRayShape, 74
GetIntersection, 75
SetPoints, 75
Update, 75
DARTBallJoint
gazebo::physics::DARTBallJoint, 281
DARTBallJoint.hh, 1221
DARTBoxShape
gazebo::physics::DARTBoxShape, 285
DARTBoxShape.hh, 1222
DARTCollision
gazebo::physics::DARTCollision, 288
DARTCollision.hh, 1222
DARTCollisionPtr
gazebo::physics, 116
DARTCylinderShape
gazebo::physics::DARTCylinderShape, 292
DARTCylinderShape.hh, 1223
DARTHeightmapShape
gazebo::physics::DARTHeightmapShape, 294
DARTHeightmapShape.hh, 1224
DARTHinge2Joint
gazebo::physics::DARTHinge2Joint, 296
DARTHinge2Joint.hh, 1224
DARTHingeJoint
gazebo::physics::DARTHingeJoint, 301
DARTHingeJoint.hh, 1225
DARTJoint
gazebo::physics::DARTJoint, 307
DARTJoint.hh, 1226
DARTJointPtr
gazebo::physics, 116
DARTLink
gazebo::physics::DARTLink, 316
DARTLink.hh, 1226
DARTLinkPtr

gazebo::physics, 116
DARTMeshShape
gazebo::physics::DARTMeshShape, 325
DARTMeshShape.hh, 1227
DARTModel
gazebo::physics::DARTModel, 327
DARTModel.hh, 1227
DARTModelPtr
gazebo::physics, 116
DARTMultiRayShape
Bullet Physics, 76
DARTMultiRayShape.hh, 1228
DARTParam
gazebo::physics::DARTPhysics, 332
DARTPhysics
gazebo::physics::DARTPhysics, 332
DARTPhysics.hh, 1229
DARTPhysicsPtr
gazebo::physics, 117
DARTPlaneShape
gazebo::physics::DARTPlaneShape, 337
DARTPlaneShape.hh, 1229
DARTRayShape
DART Physics, 74
DARTRayShape.hh, 1230
DARTRayShapePtr
gazebo::physics, 117
DARTScrewJoint
gazebo::physics::DARTScrewJoint, 341
DARTScrewJoint.hh, 1230
DARTSliderJoint
gazebo::physics::DARTSliderJoint, 347
DARTSliderJoint.hh, 1231
DARTSphereShape
gazebo::physics::DARTSphereShape, 351
DARTSphereShape.hh, 1231
DARTTypes.hh, 1232
DARTUniversalJoint
gazebo::physics::DARTUniversalJoint, 354
DARTUniversalJoint.hh, 1233
DEFERRED
gazebo::rendering::RenderEngine, 793
DIAG_TIMER_LAP
Utility, 98
DIAG_TIMER_START
Utility, 98
DIAG_TIMER_STOP
Utility, 98
DIFFERENCE
gazebo::common::MeshCSG, 614
damper
gazebo::physics::SimbodyJoint, 899
dampingCoefficient
gazebo::physics::Joint, 515
dart_inc.h, 1220
dartPhysicsEngine
gazebo::physics::DARTJoint, 313
dartScrewJoint
gazebo::physics::DARTScrewJoint, 344
DebugPrint
gazebo::physics::DARTPhysics, 333
gazebo::physics::PhysicsEngine, 711
gazebo::physics::SimbodyPhysics, 919
DebugString
gazebo::physics::Contact, 261
DecCount
gazebo::transport::IOManager, 495
Decode
gazebo::common::AudioDecoder, 154
DecodeTopicName
gazebo::transport::Node, 675
defaultVpParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 860
defxAB
gazebo::physics::SimbodyJoint, 899
Degree
gazebo::math::Angle, 140
DeleteDynamicLine
gazebo::rendering::Visual, 1129
deleteEntity
gazebo::event::Events, 403
DepthCamera
gazebo::rendering::DepthCamera, 359
DepthCamera.hh, 1234
DepthCameraPtr
gazebo::rendering, 121
DepthCameraSensor
gazebo::sensors::DepthCameraSensor, 363
DepthCameraSensor.hh, 1234
DepthCameraSensor_V
gazebo::sensors, 125
DepthCameraSensorPtr
gazebo::sensors, 125
depthTarget
gazebo::rendering::DepthCamera, 362
depthTexture
gazebo::rendering::DepthCamera, 362
depthViewport
gazebo::rendering::DepthCamera, 362
depths
gazebo::physics::Contact, 262
Detach
gazebo::physics::DARTJoint, 307
gazebo::physics::Joint, 502
gazebo::physics::SimbodyJoint, 894

Rendering, 84
DetachAllStaticModels
gazebo::physics::Link, 549
DetachEntity
gazebo::rendering::RTShaderSystem, 813
DetachObjects
gazebo::rendering::Visual, 1129
DetachStaticModel
gazebo::physics::Link, 549
gazebo::physics::Model, 628
DetachViewport
gazebo::rendering::RTShaderSystem, 813
DetachVisual
gazebo::rendering::Visual, 1129
diagTimerStart
gazebo::event::Events, 403
diagTimerStop
gazebo::event::Events, 404
DiagnosticTimer
gazebo::util::DiagnosticTimer, 369
DiagnosticTimerPtr
gazebo::common, 104
gazebo::util, 129
Diagnostics.hh, 1235
diffuse
gazebo::common::Material, 593
dirtyPose
gazebo::physics::Entity, 389
dirtyPoses
gazebo::physics::World, 1168
disable
Sensors, 89
DisableAllModels
gazebo::physics::World, 1160
DisableTrackVisual
gazebo::rendering::Visual, 1129
Disconnect
Events, 46, 47
gazebo::event::Event, 392
DisconnectAddEntity
gazebo::event::Events, 400
DisconnectCreateEntity
gazebo::event::Events, 400
DisconnectCreateScene
gazebo::rendering::Events, 406
DisconnectDeleteEntity
gazebo::event::Events, 400
DisconnectDiagTimerStart
gazebo::event::Events, 401
DisconnectDiagTimerStop
gazebo::event::Events, 401
DisconnectEnabled
gazebo::physics::Link, 549
DisconnectJointUpdate
gazebo::physics::Joint, 502
DisconnectNewDepthFrame
gazebo::rendering::DepthCamera, 360
DisconnectNewImageFrame
gazebo::rendering::Camera, 195
DisconnectNewLaserFrame
gazebo::rendering::GpuLaser, 432
gazebo::sensors::GpuRaySensor, 442
DisconnectNewLaserScans
gazebo::physics::MultiRayShape, 668
DisconnectNewRGBPointCloud
gazebo::rendering::DepthCamera, 360
DisconnectPause
gazebo::event::Events, 401
DisconnectPostRender
gazebo::event::Events, 401
DisconnectPreRender
gazebo::event::Events, 401
DisconnectPubFromSub
gazebo::transport::TopicManager, 1058
DisconnectRemoveScene
gazebo::rendering::Events, 407
DisconnectRender
gazebo::event::Events, 402
DisconnectSetSelectedEntity
gazebo::event::Events, 402
DisconnectShutdown
gazebo::transport::Connection, 251
DisconnectSigInt
gazebo::event::Events, 402
DisconnectStep
gazebo::event::Events, 402
DisconnectStop
gazebo::event::Events, 402
DisconnectSubFromPub
gazebo::transport::TopicManager, 1058
DisconnectUpdate
gazebo::sensors::ForceTorqueSensor, 420
gazebo::sensors::SonarSensor, 979
DisconnectUpdated
gazebo::sensors::Sensor, 841
DisconnectWorldCreated
gazebo::event::Events, 403
DisconnectWorldUpdateBegin
gazebo::event::Events, 403
DisconnectWorldUpdateEnd
gazebo::event::Events, 403
discreteForces
gazebo::physics::SimbodyPhysics, 924
Distance
gazebo::math::Plane, 727
gazebo::math::Vector2d, 1077
gazebo::math::Vector2i, 1085
gazebo::math::Vector3, 1095, 1096

gazebo::math::Vector4, 1108
Dot
gazebo::math::Quaternion, 765
gazebo::math::Vector3, 1096
Double
gazebo::common::Time, 1036
DownloadDependencies
Common, 39
dragging
gazebo::common::MouseEvent, 651
DrawLine
gazebo::rendering::Scene, 821
dtBallJoint
gazebo::physics::DARTBallJoint, 284
dtChildBodyNode
gazebo::physics::DARTJoint, 313
dtConfig
gazebo::physics::DARTModel, 328
dtJoint
gazebo::physics::DARTJoint, 313
dtPrismaticJoint
gazebo::physics::DARTSliderJoint, 349
dtRevoluteJoint
gazebo::physics::DARTHingeJoint, 304
dtSkeleton
gazebo::physics::DARTModel, 328
dtUniveralJoint
gazebo::physics::DARTHinge2Joint, 299
gazebo::physics::DARTUniversalJoint, 357
dtVelocity
gazebo::physics::DARTModel, 328
dummyContext
gazebo::rendering::RenderEngine, 795
dummyDisplay
gazebo::rendering::RenderEngine, 795
dummyWindowId
gazebo::rendering::RenderEngine, 795
duration
gazebo::physics::TrajectoryInfo, 1061
DynamicLines
gazebo::rendering::DynamicLines, 372
DynamicLines.hh, 1236
DynamicLinesPtr
gazebo::rendering, 121
DynamicRenderable
gazebo::rendering::DynamicRenderable, 376
DynamicRenderable.hh, 1237

EARTH_WGS84
gazebo::common::SphericalCoordinates, 990
ENTITY
gazebo::physics::Base, 163
ERP
gazebo::physics::Joint, 500
effortLimit
gazebo::physics::Joint, 515
emissive
gazebo::common::Material, 593
Enable
gazebo::rendering::Grid, 451
enable
Sensors, 90
EnableAllModels
gazebo::physics::World, 1160
EnablePhysicsEngine
gazebo::physics::World, 1160
EnableSaveFrame
gazebo::rendering::Camera, 195
EnableTrackVisual
gazebo::rendering::Visual, 1129
EnableViewController
gazebo::rendering::UserCamera, 1069
enabled
gazebo::rendering::ViewController, 1121
EncodeTopicName
gazebo::transport::Node, 675
endTime
gazebo::physics::TrajectoryInfo, 1061
EnqueueMsg
gazebo::transport::Connection, 251
Entity
gazebo::physics::Entity, 382
Entity.hh, 1237
entityCreated
gazebo::event::Events, 404
EntityPtr
gazebo::physics, 117
EntityType
gazebo::physics::Base, 163
EntityTypename
Classes for physics and dynamics, 72
Equal
gazebo::math::Vector3, 1096
equal
Math, 50
erp
gazebo::physics::SurfaceParams, 1023
EulerToQuaternion
gazebo::math::Quaternion, 765
Event.hh, 1238
eventConnections
gazebo::transport::ConnectionManager, 259
EventType
gazebo::common::KeyEvent, 532
gazebo::common::MouseEvent, 651
Events, 45
∼EventT, 45
Connect, 45

ConnectionCount, 46
Disconnect, 46, 47
Events.hh, 1239
Exception
gazebo::common::Exception, 417
Exception.hh, 1240

FACE_MAX
STLLoader.hh, 1379
FLAT
gazebo::common::Material, 588
FMAX
gazebo::physics::Joint, 500
FORWARD
gazebo::rendering::RenderEngine, 793
FPSViewController
gazebo::rendering::FPSViewController, 424
FPSViewController.hh, 1242
FUDGE_FACTOR
gazebo::physics::Joint, 500
fakeAnchor
gazebo::physics::ScrewJoint, 835
gazebo::physics::SliderJoint, 975
far
gazebo::rendering::GpuLaser, 437
fdir1
gazebo::physics::SurfaceParams, 1023
filename
gazebo::PluginT, 734
FillArrays
gazebo::common::Mesh, 609
gazebo::common::SubMesh, 1009
FillBufferFromFile
gazebo::util::OpenALSource, 700
FillBufferFromPCM
gazebo::util::OpenALSource, 700
FillHardwareBuffers
gazebo::rendering::DynamicRenderable, 376
FillMsg
gazebo::physics::BoxShape, 178
gazebo::physics::Collision, 223
gazebo::physics::Contact, 261
gazebo::physics::CylinderShape, 277
gazebo::physics::HeightmapShape, 468
gazebo::physics::Joint, 502
gazebo::physics::Link, 550
gazebo::physics::MapShape, 581
gazebo::physics::MeshShape, 622
gazebo::physics::Model, 629
gazebo::physics::MultiRayShape, 668
gazebo::physics::PlaneShape, 730
gazebo::physics::RayShape, 788
gazebo::physics::Shape, 863
gazebo::physics::SphereShape, 987
gazebo::physics::SurfaceParams, 1022
gazebo::rendering::Light, 537
gazebo::sensors::Sensor, 841
FillSDF
gazebo::physics::CollisionState, 231
gazebo::physics::JointState, 525
gazebo::physics::LinkState, 566
gazebo::physics::ModelState, 644
gazebo::physics::WorldState, 1173
find_file
Common, 40
gazebo, 101
find_file_path
Common, 40
FindFile
gazebo::common::SystemPaths, 1027
FindFileURI
gazebo::common::SystemPaths, 1028
FindPublication
gazebo::transport::TopicManager, 1058
Fini
Common, 40
gazebo::Master, 584
gazebo::physics::Actor, 134
gazebo::physics::Base, 164
gazebo::physics::Collision, 223
gazebo::physics::DARTCollision, 288
gazebo::physics::DARTLink, 318
gazebo::physics::DARTModel, 327
gazebo::physics::DARTPhysics, 333
gazebo::physics::Entity, 382
gazebo::physics::Link, 550
gazebo::physics::Model, 629
gazebo::physics::PhysicsEngine, 712
gazebo::physics::SimbodyLink, 905
gazebo::physics::SimbodyPhysics, 919
gazebo::physics::World, 1161
gazebo::rendering::Camera, 195
gazebo::rendering::DepthCamera, 361
gazebo::rendering::GpuLaser, 433
gazebo::rendering::RenderEngine, 794
gazebo::rendering::RTShaderSystem, 813
gazebo::rendering::UserCamera, 1069
gazebo::rendering::Visual, 1130
gazebo::rendering::WindowManager, 1145
gazebo::sensors::CameraSensor, 215
gazebo::sensors::ContactSensor, 269
gazebo::sensors::DepthCameraSensor, 363
gazebo::sensors::ForceTorqueSensor, 421
gazebo::sensors::GpsSensor, 427
gazebo::sensors::GpuRaySensor, 442
gazebo::sensors::ImuSensor, 481
gazebo::sensors::MultiCameraSensor, 661
gazebo::sensors::RaySensor, 781

gazebo::sensors::RFIDSensor, 797
gazebo::sensors::RFIDTag, 800
gazebo::sensors::Sensor, 841
gazebo::sensors::SensorManager, 851
gazebo::sensors::SonarSensor, 980
gazebo::sensors::WirelessReceiver, 1149
gazebo::sensors::WirelessTransceiver, 1152
gazebo::Server, 856
gazebo::transport::ConnectionManager, 256
gazebo::transport::Node, 675
gazebo::transport::PublicationTransport, 756
gazebo::transport::TopicManager, 1058
gazebo::util::LogRecord, 576
gazebo::util::OpenAL, 697
fini
Classes for physics and dynamics, 69
gazebo, 101
Rendering, 84
Sensors, 90
Transport, 94
fixnan
Math, 50
Flatten
gazebo::rendering::Heightmap, 462
flipY
gazebo::physics::HeightmapShape, 470
Float
gazebo::common::Time, 1036
FogFromSDF
Messages, 59
ForceTorque1
Joint_TEST, 518
ForceTorque2
Joint_TEST, 518
ForceTorqueSensor
gazebo::sensors::ForceTorqueSensor, 420
ForceTorqueSensor.hh, 1242
ForceTorqueSensorPtr
gazebo::sensors, 125
forces
gazebo::physics::SimbodyPhysics, 924
freq
gazebo::sensors::WirelessTransmitter, 1156

g
gazebo::common::Color, 244
GAUSSIAN
gazebo::sensors::Noise, 690
GAUSSIAN_QUANTIZED
gazebo::sensors::Noise, 690
GAZEBO_DEPRECATED
CommonTypes.hh, 1208
GAZEBO_FORCEINLINE
CommonTypes.hh, 1208
GOURAUD
gazebo::common::Material, 588
GPtrArray
MeshCSG.hh, 1283
GUIFromSDF
Messages, 60
GUIOverlay
gazebo::rendering::GUIOverlay, 456
GUIOverlay.hh, 1250
GUIPluginPtr
gazebo, 101
GZ_ALL_COLLIDE
PhysicsTypes.hh, 1310
GZ_ASSERT
Assert.hh, 1186
GZ_DBL_MAX
Helpers.hh, 1254
GZ_DBL_MIN
Helpers.hh, 1254
GZ_DTOR
Angle.hh, 1182
GZ_FIXED_COLLIDE
PhysicsTypes.hh, 1310
GZ_FLT_MAX
Helpers.hh, 1254
GZ_FLT_MIN
Helpers.hh, 1254
GZ_GHOST_COLLIDE
PhysicsTypes.hh, 1310
GZ_LOG_VERSION
LogRecord.hh, 1274
GZ_MODEL_DB_MANIFEST_FILENAME
ModelDatabase.hh, 1289
GZ_MODEL_MANIFEST_FILENAME
ModelDatabase.hh, 1289
GZ_NONE_COLLIDE
PhysicsTypes.hh, 1310
GZ_NORMALIZE
Angle.hh, 1182
GZ_REGISTER_MODEL_PLUGIN
Plugin.hh, 1315
GZ_REGISTER_PHYSICS_ENGINE
Classes for physics and dynamics, 69
GZ_REGISTER_SENSOR_PLUGIN
Plugin.hh, 1316
GZ_REGISTER_STATIC_MSG
Messages, 56
GZ_REGISTER_STATIC_SENSOR
Sensors, 89
GZ_REGISTER_SYSTEM_PLUGIN
Plugin.hh, 1316
GZ_REGISTER_VISUAL_PLUGIN
Plugin.hh, 1316
GZ_REGISTER_WORLD_PLUGIN

Plugin.hh, 1317
GZ_RTOD
Angle.hh, 1182
GZ_SENSOR_COLLIDE
PhysicsTypes.hh, 1310
GZ_SKYX_ALL
gazebo::rendering::Scene, 818
GZ_SKYX_CLOUDS
gazebo::rendering::Scene, 818
GZ_SKYX_MOON
gazebo::rendering::Scene, 818
GZ_SKYX_NONE
gazebo::rendering::Scene, 818
GZ_UINT32_MAX
Helpers.hh, 1254
GZ_UINT32_MIN
Helpers.hh, 1255
GZ_VISIBILITY_ALL
RenderTypes.hh, 1333
GZ_VISIBILITY_GUI
RenderTypes.hh, 1333
GZ_VISIBILITY_SELECTABLE
RenderTypes.hh, 1334
GZ_VISIBILITY_SELECTION
RenderTypes.hh, 1334
gain
gazebo::sensors::WirelessTransceiver, 1153
gazebo, 99
add_plugin, 101
find_file, 101
fini, 101
GUIPluginPtr, 101
init, 101
load, 101
ModelPluginPtr, 101
print_version, 101
run, 101
SensorPluginPtr, 101
stop, 101
SystemPluginPtr, 101
VisualPluginPtr, 101
WorldPluginPtr, 101
gazebo.hh, 1243
gazebo::Master, 583
∼Master, 584
Fini, 584
Init, 584
Master, 584
Run, 584
RunOnce, 584
RunThread, 584
Stop, 585
gazebo::ModelPlugin, 639
∼ModelPlugin, 640
Init, 641
Load, 641
ModelPlugin, 640
Reset, 641
gazebo::PluginT
∼PluginT, 733
Create, 733
filename, 734
GetFilename, 734
GetHandle, 734
GetType, 734
handle, 734
PluginT, 733
TPtr, 733
type, 734
gazebo::PluginT< T >, 732
gazebo::SensorPlugin, 853
∼SensorPlugin, 854
Init, 855
Load, 855
Reset, 855
SensorPlugin, 854
gazebo::Server, 855
∼Server, 856
Fini, 856
GetInitialized, 856
Init, 856
LoadFile, 856
LoadString, 856
ParseArgs, 856
PreLoad, 856
PrintUsage, 856
Run, 856
Server, 856
SetParams, 856
Stop, 856
systemPluginsArgc, 856
systemPluginsArgv, 857
gazebo::SystemPlugin, 1030
∼SystemPlugin, 1031
Init, 1031
Load, 1031
Reset, 1031
SystemPlugin, 1030
gazebo::VisualPlugin, 1142
Init, 1143
Load, 1143
Reset, 1143
VisualPlugin, 1143
gazebo::WorldPlugin, 1169
∼WorldPlugin, 1170
Init, 1170
Load, 1170
Reset, 1170

WorldPlugin, 1170
gazebo::common, 101
AnimationPtr, 104
DiagnosticTimerPtr, 104
NodeMap, 104
NodeMapIter, 104
NumericAnimationPtr, 104
Param_V, 104
PoseAnimationPtr, 104
RawNodeAnim, 104
RawNodeWeights, 104
RawSkeletonAnim, 104
SpeedOfLight, 105
SphericalCoordinatesPtr, 104
StrStr_M, 105
gazebo::common::Animation, 145
∼Animation, 147
AddTime, 147
Animation, 147
build, 149
GetKeyFrame, 147
GetKeyFrameCount, 147
GetKeyFramesAtTime, 148
GetLength, 148
GetTime, 148
KeyFrame_V, 147
keyFrames, 149
length, 149
loop, 149
name, 149
SetLength, 148
SetTime, 148
timePos, 149
gazebo::common::AssertionInternalError, 151
∼AssertionInternalError, 153
AssertionInternalError, 152
gazebo::common::AudioDecoder, 153
∼AudioDecoder, 153
AudioDecoder, 153
Decode, 154
GetFile, 154
GetSampleRate, 154
SetFile, 154
gazebo::common::BVHLoader, 179
∼BVHLoader, 180
BVHLoader, 180
Load, 180
gazebo::common::ColladaLoader, 218
∼ColladaLoader, 219
ColladaLoader, 219
Load, 219
gazebo::common::Color, 233
∼Color, 236
a, 244
ABGR, 236
ARGB, 236
b, 244
BGRA, 236
Black, 244
Blue, 244
Color, 236
g, 244
GetAsABGR, 237
GetAsARGB, 237
GetAsBGRA, 237
GetAsHSV, 237
GetAsRGBA, 237
GetAsYUV, 237
Green, 244
operator<<, 243
operator>>, 243
operator∗, 238
operator∗=, 238
operator+, 239
operator+=, 239
operator-, 239, 240
operator-=, 240
operator/, 240
operator/=, 241
operator=, 241
operator==, 241
operator[], 241
Purple, 244
r, 244
RGBA, 236
Red, 244
Reset, 242
Set, 242
SetFromABGR, 242
SetFromARGB, 242
SetFromBGRA, 242
SetFromHSV, 242
SetFromRGBA, 243
SetFromYUV, 243
White, 244
Yellow, 244
gazebo::common::Console, 259
gazebo::common::Exception, 416
∼Exception, 417
Exception, 417
GetErrorFile, 417
GetErrorStr, 418
operator<<, 418
Print, 418
gazebo::common::Image, 474
∼Image, 476
BAYER_GBRG8, 476
BAYER_GRBG8, 476

BAYER_RGGB8, 476
BAYER_RGGR8, 476
BGR_INT16, 476
BGR_INT32, 476
BGR_INT8, 476
BGRA_INT8, 476
ConvertPixelFormat, 476
GetAvgColor, 477
GetBPP, 477
GetData, 477
GetFilename, 477
GetHeight, 477
GetMaxColor, 478
GetPitch, 478
GetPixel, 478
GetPixelFormat, 478
GetRGBData, 478
GetWidth, 478
Image, 476
L_INT16, 476
L_INT8, 476
Load, 479
PIXEL_FORMAT_COUNT, 476
PixelFormat, 476
R_FLOAT16, 476
R_FLOAT32, 476
RGB_FLOAT16, 476
RGB_FLOAT32, 476
RGB_INT16, 476
RGB_INT32, 476
RGB_INT8, 476
RGBA_INT8, 476
Rescale, 479
SavePNG, 479
SetFromData, 479
UNKNOWN_PIXEL_FORMAT, 476
Valid, 479
gazebo::common::InternalError, 493
~InternalError, 494
InternalError, 494
gazebo::common::KeyEvent, 531
EventType, 532
key, 532
KeyEvent, 532
NO_EVENT, 532
PRESS, 532
RELEASE, 532
type, 532
gazebo::common::KeyFrame, 532
~KeyFrame, 533
GetTime, 533
KeyFrame, 533
time, 533
gazebo::common::Material, 585
~Material, 588
ADD, 587
ambient, 593
BLEND_COUNT, 587
BLINN, 588
BlendMode, 587
blendMode, 593
BlendModeStr, 593
diffuse, 593
emissive, 593
FLAT, 588
GOURAUD, 588
GetAmbient, 588
GetBlendFactors, 588
GetBlendMode, 588
GetDepthWrite, 589
GetDiffuse, 589
GetEmissive, 589
GetLighting, 589
GetName, 589
GetPointSize, 589
GetShadeMode, 590
GetShininess, 590
GetSpecular, 590
GetTextureImage, 590
GetTransparency, 590
MODULATE, 587
Material, 588
name, 593
operator<<, 593
PHONG, 588
pointSize, 593
REPLACE, 587
SHADE_COUNT, 588
SetAmbient, 590
SetBlendFactors, 591
SetBlendMode, 591
SetDepthWrite, 591
SetDiffuse, 591
SetEmissive, 591
SetLighting, 592
SetPointSize, 592
SetShadeMode, 592
SetShininess, 592
SetSpecular, 592
SetTextureImage, 592
SetTransparency, 593
ShadeMode, 587
shadeMode, 594
ShadeModeStr, 594
shininess, 594
specular, 594
texImage, 594
transparency, 594

gazebo::common::Mesh, 606
~Mesh, 608
AddMaterial, 608
AddSubMesh, 608
Center, 608
FillArrays, 609
GenSphericalTexCoord, 609
GetAABB, 609
GetIndexCount, 609
GetMaterial, 609
GetMaterialCount, 610
GetMax, 610
GetMin, 610
GetName, 610
GetNormalCount, 610
GetPath, 610
GetSkeleton, 610
GetSubMesh, 611
GetSubMeshCount, 611
GetTexCoordCount, 611
GetVertexCount, 611
HasSkeleton, 612
Mesh, 608
RecalculateNormals, 612
Scale, 612
SetName, 612
SetPath, 612
SetScale, 612
SetSkeleton, 612
Translate, 613
gazebo::common::MeshCSG, 613
~MeshCSG, 614
BooleanOperation, 613
CreateBoolean, 614
DIFFERENCE, 614
INTERSECTION, 614
MeshCSG, 614
UNION, 614
gazebo::common::MeshLoader, 614
~MeshLoader, 615
Load, 615
MeshLoader, 615
gazebo::common::MeshManager, 616
AddMesh, 617
CreateBox, 617
CreateCamera, 617
CreateCone, 618
CreateCylinder, 618
CreatePlane, 618
CreateSphere, 619
CreateTube, 619
GenSphericalTexCoord, 619
GetMesh, 619
GetMeshAABB, 620
HasMesh, 620
IsValidFilename, 620
Load, 620
gazebo::common::ModelDatabase, 638
gazebo::common::MouseEvent, 649
alt, 651
button, 651
Buttons, 651
buttons, 651
control, 651
dragging, 651
EventType, 651
LEFT, 651
MIDDLE, 651
MOVE, 651
MouseEvent, 651
moveScale, 652
NO_BUTTON, 651
NO_EVENT, 651
PRESS, 651
pos, 652
pressPos, 652
prevPos, 652
RELEASE, 651
RIGHT, 651
SCROLL, 651
scroll, 652
shift, 652
type, 652
gazebo::common::NodeAnimation, 679
~NodeAnimation, 681
AddKeyFrame, 681
GetFrameAt, 681
GetFrameCount, 681
GetKeyFrame, 681, 682
GetLength, 682
GetName, 682
GetTimeAtX, 682
keyFrames, 683
length, 683
name, 683
NodeAnimation, 680
Scale, 682
SetName, 683
gazebo::common::NodeAssignment, 683
nodeIndex, 684
vertexIndex, 684
weight, 684
gazebo::common::NodeTransform, 684
~NodeTransform, 686
Get, 686
GetSID, 686
GetType, 687
MATRIX, 686

NodeTransform, 686
operator∗, 687
operator(), 687
PrintSource, 687
ROTATE, 686
RecalculateMatrix, 688
SCALE, 686
Set, 688
SetComponent, 688
SetSID, 688
SetSourceValues, 688
SetType, 689
sid, 689
source, 689
TRANSLATE, 686
transform, 689
TransformType, 686
type, 689
gazebo::common::NumericAnimation, 692
∼NumericAnimation, 693
CreateKeyFrame, 693
GetInterpolatedKeyFrame, 693
NumericAnimation, 692
gazebo::common::NumericKeyFrame, 693
∼NumericKeyFrame, 694
GetValue, 695
NumericKeyFrame, 694
SetValue, 695
value, 695
gazebo::common::PID, 722
∼PID, 723
GetCmd, 723
GetErrors, 723
Init, 723
operator=, 724
PID, 723
Reset, 724
SetCmd, 724
SetCmdMax, 724
SetCmdMin, 724
SetDGain, 725
SetIGain, 725
SetIMax, 725
SetIMin, 725
SetPGain, 725
Update, 725
gazebo::common::ParamT< T >, 706
gazebo::common::PoseAnimation, 743
∼PoseAnimation, 745
BuildInterpolationSplines, 745
CreateKeyFrame, 745
GetInterpolatedKeyFrame, 745
PoseAnimation, 744
gazebo::common::PoseKeyFrame, 746
∼PoseKeyFrame, 747
GetRotation, 747
GetTranslation, 747
PoseKeyFrame, 747
rotate, 748
SetRotation, 747
SetTranslation, 747
translate, 748
gazebo::common::STLLoader, 1002
∼STLLoader, 1003
Load, 1003
STLLoader, 1003
gazebo::common::Skeleton, 953
∼Skeleton, 955
AddAnimation, 955
AddVertNodeWeight, 955
anims, 959
bindShapeTransform, 959
BuildNodeMap, 956
GetAnimation, 956
GetBindShapeTransform, 956
GetNodeByHandle, 956
GetNodeById, 956
GetNodeByName, 957
GetNodes, 957
GetNumAnimations, 957
GetNumJoints, 957
GetNumNodes, 957
GetNumVertNodeWeights, 957
GetRootNode, 958
GetVertNodeWeight, 958
nodes, 959
PrintTransforms, 958
rawNW, 959
root, 959
Scale, 958
SetBindShapeTransform, 958
SetNumVertAttached, 959
SetRootNode, 959
Skeleton, 955
gazebo::common::SkeletonAnimation, 960
∼SkeletonAnimation, 961
AddKeyFrame, 961
animations, 963
GetLength, 961
GetName, 962
GetNodeCount, 962
GetNodePoseAt, 962
GetPoseAt, 962
GetPoseAtX, 963
HasNode, 963
length, 963
name, 964
Scale, 963

SetName, 963
SkeletonAnimation, 961
gazebo::common::SkeletonNode, 964
~SkeletonNode, 967
AddChild, 967
AddRawTransform, 967
children, 972
GetChild, 967
GetChildById, 967
GetChildByName, 968
GetChildCount, 968
GetHandle, 968
GetId, 968
GetInverseBindTransform, 968
GetModelTransform, 968
GetName, 969
GetNumRawTrans, 969
GetParent, 969
GetRawTransform, 969
GetRawTransforms, 969
GetTransform, 970
GetTransforms, 970
handle, 972
id, 972
initialTransform, 972
invBindTransform, 972
IsJoint, 970
IsRootNode, 970
JOINT, 966
modelTransform, 972
NODE, 966
name, 972
parent, 973
rawTransforms, 973
Reset, 970
SetHandle, 970
SetId, 970
SetInitialTransform, 971
SetInverseBindTransform, 971
SetModelTransform, 971
SetName, 971
SetParent, 971
SetTransform, 971
SetType, 972
SkeletonNode, 966, 967
SkeletonNodeType, 966
transform, 973
type, 973
UpdateChildrenTransforms, 972
gazebo::common::SphericalCoordinates, 988
~SphericalCoordinates, 990
Convert, 991
EARTH_WGS84, 990
GetElevationReference, 991
GetHeadingOffset, 991
GetLatitudeReference, 991
GetLongitudeReference, 991
GetSurfaceType, 991
GlobalFromLocal, 992
SetElevationReference, 992
SetHeadingOffset, 992
SetLatitudeReference, 992
SetLongitudeReference, 992
SetSurfaceType, 993
SphericalCoordinates, 990
SphericalFromLocal, 993
SurfaceType, 990
gazebo::common::SubMesh, 1004
~SubMesh, 1007
AddIndex, 1007
AddNodeAssignment, 1007
AddNormal, 1007
AddTexCoord, 1007
AddVertex, 1008
Center, 1008
CopyNormals, 1008
CopyVertices, 1008
FillArrays, 1009
GenSphericalTexCoord, 1009
GetIndex, 1009
GetIndexCount, 1009
GetMaterialIndex, 1009
GetMax, 1009
GetMaxIndex, 1009
GetMin, 1009
GetName, 1010
GetNodeAssignment, 1010
GetNodeAssignmentsCount, 1010
GetNormal, 1010
GetNormalCount, 1010
GetPrimitiveType, 1010
GetTexCoord, 1011
GetTexCoordCount, 1011
GetVertex, 1011
GetVertexCount, 1011
GetVertexIndex, 1011
HasVertex, 1011
LINES, 1006
LINESTRIPS, 1006
POINTS, 1006
PrimitiveType, 1006
RecalculateNormals, 1012
Scale, 1012
SetIndexCount, 1012
SetMaterialIndex, 1012
SetName, 1012
SetNormal, 1012
SetNormalCount, 1013

SetPrimitiveType, 1013
SetScale, 1013
SetSubMeshCenter, 1013
SetTexCoord, 1013
SetTexCoordCount, 1013
SetVertex, 1014
SetVertexCount, 1014
SubMesh, 1006
TRIANGLES, 1006
TRIFANS, 1006
TRISTRIPS, 1006
Translate, 1014
gazebo::common::SystemPaths, 1025
AddGazeboPaths, 1026
AddModelPaths, 1026
AddOgrePaths, 1027
AddPluginPaths, 1027
AddSearchPathSuffix, 1027
ClearGazeboPaths, 1027
ClearModelPaths, 1027
ClearOgrePaths, 1027
ClearPluginPaths, 1027
FindFile, 1027
FindFileURI, 1028
gazeboPathsFromEnv, 1029
GetGazeboPaths, 1028
GetLogPath, 1028
GetModelPaths, 1028
GetOgrePaths, 1028
GetPluginPaths, 1029
GetWorldPathExtension, 1029
modelPathsFromEnv, 1029
ogrePathsFromEnv, 1029
pluginPathsFromEnv, 1029
gazebo::common::Time, 1031
∼Time, 1036
Double, 1036
Float, 1036
GetWallTime, 1036
GetWallTimeAsISOString, 1037
MSleep, 1037
MicToNano, 1037
MilToNano, 1037
NSleep, 1038
nsec, 1052
operator<, 1045, 1046
operator<<, 1052
operator<=, 1046, 1047
operator>, 1049
operator>>, 1052
operator>=, 1050
operator∗, 1039
operator∗=, 1040
operator+, 1040, 1041
operator+=, 1041, 1042
operator-, 1042
operator-=, 1043
operator/, 1043, 1044
operator/=, 1044, 1045
operator=, 1047
operator==, 1048
sec, 1052
SecToNano, 1051
Set, 1051
SetToWallTime, 1051
Sleep, 1051
Time, 1035, 1036
Zero, 1052
gazebo::common::Timer, 1052
∼Timer, 1053
GetElapsed, 1054
GetRunning, 1054
operator<<, 1054
Start, 1054
Stop, 1054
Timer, 1053
gazebo::common::UpdateInfo, 1065
realTime, 1065
simTime, 1065
worldName, 1066
gazebo::common::Video, 1115
∼Video, 1115
GetHeight, 1115
GetNextFrame, 1115
GetWidth, 1116
Load, 1116
Video, 1115
gazebo::event, 105
Connection_V, 105
ConnectionPtr, 105
gazebo::event::Connection, 246
∼Connection, 247
Connection, 247
GetId, 247
gazebo::event::Event, 390
∼Event, 392
Disconnect, 392
gazebo::event::EventT
operator(), 410–412
Signal, 413–415
gazebo::event::EventT< T >, 407
gazebo::event::Events, 393
addEntity, 403
ConnectAddEntity, 395
ConnectCreateEntity, 396
ConnectDeleteEntity, 396
ConnectDiagTimerStart, 396
ConnectDiagTimerStop, 397

ConnectPause, 397
ConnectPostRender, 397
ConnectPreRender, 397
ConnectRender, 398
ConnectSetSelectedEntity, 398
ConnectSigInt, 398
ConnectStep, 399
ConnectStop, 399
ConnectWorldCreated, 399
ConnectWorldUpdateBegin, 399
ConnectWorldUpdateEnd, 400
deleteEntity, 403
diagTimerStart, 403
diagTimerStop, 404
DisconnectAddEntity, 400
DisconnectCreateEntity, 400
DisconnectDeleteEntity, 400
DisconnectDiagTimerStart, 401
DisconnectDiagTimerStop, 401
DisconnectPause, 401
DisconnectPostRender, 401
DisconnectPreRender, 401
DisconnectRender, 402
DisconnectSetSelectedEntity, 402
DisconnectSigInt, 402
DisconnectStep, 402
DisconnectStop, 402
DisconnectWorldCreated, 403
DisconnectWorldUpdateBegin, 403
DisconnectWorldUpdateEnd, 403
entityCreated, 404
pause, 404
postRender, 404
preRender, 404
render, 404
setSelectedEntity, 404
sigInt, 404
step, 404
stop, 405
worldCreated, 405
worldUpdateBegin, 405
worldUpdateEnd, 405
gazebo::math, 105
GeneratorType, 108
NRealGen, 108
NormalRealDist, 108
UIntGen, 108
URealGen, 108
UniformIntDist, 108
UniformRealDist, 108
gazebo::math::Angle, 137
∼Angle, 139
Angle, 139
Degree, 140
HalfPi, 145
Normalize, 140
operator<, 142
operator<<, 144
operator<=, 142
operator>, 143
operator>>, 144
operator>=, 143
operator∗, 140
operator∗=, 140
operator+, 141
operator+=, 141
operator-, 141
operator-=, 141
operator/, 142
operator/=, 142
operator==, 143
Pi, 145
Radian, 143
SetFromDegree, 144
SetFromRadian, 144
TwoPi, 145
Zero, 145
gazebo::math::Box, 172
∼Box, 173
Box, 173
GetCenter, 173
GetSize, 173
GetXLength, 174
GetYLength, 174
GetZLength, 174
max, 176
Merge, 174
min, 176
operator<<, 176
operator+, 174
operator+=, 175
operator-, 175
operator=, 175
operator==, 175
gazebo::math::Matrix3, 594
∼Matrix3, 596
m, 599
Matrix3, 595, 596
operator<<, 598
operator∗, 596, 598
operator+, 596
operator-, 597
operator==, 597
operator[], 597
SetCol, 597
SetFromAxes, 598
SetFromAxis, 598
gazebo::math::Matrix4, 599

~Matrix4, 601
GetAsPose, 601
GetEulerRotation, 602
GetRotation, 602
GetTranslation, 602
IDENTITY, 606
Inverse, 602
IsAffine, 602
m, 606
Matrix4, 601
operator<<, 605
operator*, 602, 603
operator=, 603
operator==, 604
operator[], 604
Set, 604
SetScale, 605
SetTranslate, 605
TransformAffine, 605
ZERO, 606
gazebo::math::Plane, 726
~Plane, 727
d, 728
Distance, 727
normal, 728
operator=, 727
Plane, 727
Set, 728
size, 728
gazebo::math::Pose, 734
~Pose, 737
CoordPoseSolve, 737
CoordPositionAdd, 737, 738
CoordPositionSub, 738
CoordRotationAdd, 738
CoordRotationSub, 738
Correct, 739
GetInverse, 739
IsFinite, 739
operator<<, 742
operator>>, 743
operator*, 739
operator+, 740
operator+=, 740
operator-, 740
operator-=, 741
operator=, 741
operator==, 741
pos, 743
Pose, 736, 737
Reset, 741
rot, 743
RotatePositionAboutOrigin, 741
Round, 742
Set, 742
Zero, 743
gazebo::math::Quaternion, 761
~Quaternion, 765
Correct, 765
Dot, 765
EulerToQuaternion, 765
GetAsAxis, 765
GetAsEuler, 766
GetAsMatrix3, 766
GetAsMatrix4, 766
GetExp, 766
GetInverse, 766
GetLog, 766
GetPitch, 767
GetRoll, 767
GetXAxis, 767
GetYAxis, 767
GetYaw, 767
GetZAxis, 767
Invert, 768
IsFinite, 768
Normalize, 768
operator<<, 773
operator>>, 774
operator*, 768, 769
operator*=, 769
operator+, 769
operator+=, 769
operator-, 770
operator-=, 770
operator=, 770
operator==, 771
Quaternion, 764
RotateVector, 771
RotateVectorReverse, 771
Round, 771
Scale, 771
Set, 772
SetFromAxis, 772
SetFromEuler, 772
SetToIdentity, 773
Slerp, 773
Squad, 773
w, 774
x, 774
y, 774
z, 774
gazebo::math::Rand, 775
GetDblNormal, 775
GetDblUniform, 775
GetIntNormal, 775
GetIntUniform, 776
GetSeed, 776

SetSeed, 776
gazebo::math::RotationSpline, 806
~RotationSpline, 807
AddPoint, 808
autoCalc, 810
Clear, 808
GetNumPoints, 808
GetPoint, 808
Interpolate, 808, 809
points, 810
RecalcTangents, 809
RotationSpline, 807
SetAutoCalculate, 809
tangents, 810
UpdatePoint, 809
gazebo::math::Spline, 993
~Spline, 994
AddPoint, 995
autoCalc, 997
Clear, 995
coeffs, 997
GetPoint, 995
GetPointCount, 995
GetTangent, 995
GetTension, 995
Interpolate, 996
points, 997
RecalcTangents, 996
SetAutoCalculate, 996
SetTension, 996
Spline, 994
tangents, 997
tension, 997
UpdatePoint, 997
gazebo::math::Vector2d, 1074
~Vector2d, 1076
Cross, 1077
Distance, 1077
IsFinite, 1077
Normalize, 1077
operator<<, 1082
operator>>, 1082
operator*, 1077, 1078
operator*=, 1078
operator+, 1079
operator+=, 1079
operator-, 1079
operator-=, 1079
operator/, 1079, 1080
operator/=, 1080
operator=, 1081
operator==, 1081
operator[], 1081
Set, 1082
Vector2d, 1076
x, 1082
y, 1082
gazebo::math::Vector2i, 1083
~Vector2i, 1085
Cross, 1085
Distance, 1085
IsFinite, 1085
Normalize, 1086
operator<<, 1091
operator>>, 1091
operator*, 1086
operator*=, 1087
operator+, 1087
operator+=, 1087
operator-, 1088
operator-=, 1088
operator/, 1088
operator/=, 1089
operator=, 1089, 1090
operator==, 1090
operator[], 1090
Set, 1090
Vector2i, 1084, 1085
x, 1091
y, 1091
gazebo::math::Vector3, 1091
~Vector3, 1095
Correct, 1095
Cross, 1095
Distance, 1095, 1096
Dot, 1096
Equal, 1096
GetAbs, 1096
GetDistToLine, 1096
GetLength, 1097
GetMax, 1097
GetMin, 1097
GetNormal, 1097
GetPerpendicular, 1097
GetRounded, 1098
GetSquaredLength, 1098
GetSum, 1098
IsFinite, 1098
Normalize, 1098
One, 1104
operator<<, 1104
operator>>, 1104
operator*, 1099, 1104
operator*=, 1099
operator+, 1100
operator+=, 1100
operator-, 1100
operator-=, 1101

operator/, 1101
operator/=, 1101
operator=, 1102
operator==, 1102
operator[], 1102
Round, 1103
Set, 1103
SetToMax, 1103
SetToMin, 1103
UnitX, 1104
UnitY, 1104
UnitZ, 1105
Vector3, 1094, 1095
x, 1105
y, 1105
z, 1105
Zero, 1105
gazebo::math::Vector4, 1105
∼Vector4, 1108
Distance, 1108
GetLength, 1108
GetSquaredLength, 1108
IsFinite, 1108
Normalize, 1108
operator<<, 1113
operator>>, 1114
operator∗, 1109
operator∗=, 1110
operator+, 1110
operator+=, 1110
operator-, 1111
operator-=, 1111
operator/, 1111
operator/=, 1112
operator=, 1112, 1113
operator==, 1113
operator[], 1113
Set, 1113
Vector4, 1107
w, 1114
x, 1114
y, 1114
z, 1114
gazebo::msgs, 108
MsgFactoryFn, 110
gazebo::msgs::MsgFactory, 658
GetMsgTypes, 659
NewMsg, 659
RegisterMsg, 659
gazebo::physics, 110
Actor_V, 116
ActorPtr, 116
Base_V, 116
BasePtr, 116
BoxShapePtr, 116
Collision_V, 116
CollisionPtr, 116
ContactPtr, 116
CylinderShapePtr, 116
DARTCollisionPtr, 116
DARTJointPtr, 116
DARTLinkPtr, 116
DARTModelPtr, 116
DARTPhysicsPtr, 117
DARTRayShapePtr, 117
EntityPtr, 117
GripperPtr, 117
HeightmapShapePtr, 117
InertialPtr, 117
Joint_V, 117
JointController_V, 117
JointControllerPtr, 117
JointPtr, 117
JointState_M, 117
Link_V, 117
LinkPtr, 117
LinkState_M, 117
MeshShapePtr, 117
Model_V, 117
ModelPtr, 117
ModelState_M, 117
MultiRayShapePtr, 117
PhysicsEnginePtr, 117
RayShapePtr, 117
RoadPtr, 117
ShapePtr, 117
SimbodyCollisionPtr, 117
SimbodyLinkPtr, 117
SimbodyModelPtr, 117
SimbodyPhysicsPtr, 118
SimbodyRayShapePtr, 118
SphereShapePtr, 118
SurfaceParamsPtr, 118
WorldPtr, 118
gazebo::physics::Actor, 131
∼Actor, 134
active, 135
Actor, 133
autoStart, 135
bonePosePub, 135
Fini, 134
GetSDF, 134
Init, 134
interpolateX, 135
IsActive, 134
lastPos, 135
lastScriptTime, 135
lastTraj, 135

Load, 134
loop, 136
mainLink, 136
mesh, 136
oldAction, 136
pathLength, 136
Play, 134
playStartTime, 136
prevFrameTime, 136
scriptLength, 136
skelAnimation, 136
skelNodesMap, 136
skeleton, 136
skinFile, 137
skinScale, 137
startDelay, 137
Stop, 134
trajInfo, 137
trajectories, 137
Update, 135
UpdateParameters, 135
visualId, 137
visualName, 137
gazebo::physics::BallJoint
∼BallJoint, 158
BallJoint, 158
GetAngleCount, 158
GetHighStop, 158
GetLowStop, 158
Load, 158
SetAxis, 159
SetHighStop, 159
SetLowStop, 159
gazebo::physics::BallJoint< T >, 157
gazebo::physics::Base, 159
∼Base, 164
ACTOR, 163
AddChild, 164
AddType, 164
BALL_JOINT, 163
BASE, 163
BOX_SHAPE, 163
Base, 164
COLLISION, 163
CYLINDER_SHAPE, 163
children, 171
childrenEnd, 171
ComputeScopedName, 164
ENTITY, 163
EntityType, 163
Fini, 164
GetByName, 165
GetChild, 165
GetChildCount, 165
GetId, 166
GetName, 166
GetParent, 166
GetParentId, 166
GetSDF, 167
GetSaveable, 166
GetScopedName, 166
GetType, 167
GetWorld, 167
HEIGHTMAP_SHAPE, 163
HINGE2_JOINT, 163
HINGE_JOINT, 163
HasType, 167
Init, 167
IsSelected, 168
JOINT, 163
LIGHT, 163
LINK, 163
Load, 168
MAP_SHAPE, 163
MESH_SHAPE, 164
MODEL, 163
MULTIRAY_SHAPE, 163
operator==, 168
PLANE_SHAPE, 164
parent, 171
Print, 169
RAY_SHAPE, 164
RemoveChild, 169
RemoveChildren, 169
Reset, 169
SCREW_JOINT, 163
SENSOR_COLLISION, 164
SHAPE, 163
SLIDER_JOINT, 163
SPHERE_SHAPE, 164
sdf, 171
SetName, 170
SetParent, 170
SetSaveable, 170
SetSelected, 170
SetWorld, 170
UNIVERSAL_JOINT, 163
Update, 171
UpdateParameters, 171
VISUAL, 163
world, 171
gazebo::physics::BoxShape, 176
∼BoxShape, 178
BoxShape, 178
FillMsg, 178
GetSize, 178
Init, 178
ProcessMsg, 178

SetScale, 179
SetSize, 179
gazebo::physics::Collision, 220
~Collision, 222
AddContact, 223
Collision, 222
FillMsg, 223
Fini, 223
GetBoundingBox, 223
GetContactsEnabled, 223
GetLaserRetro, 223
GetLink, 224
GetMaxContacts, 224
GetModel, 224
GetRelativeAngularAccel, 224
GetRelativeAngularVel, 224
GetRelativeLinearAccel, 224
GetRelativeLinearVel, 225
GetShape, 225
GetShapeType, 225
GetState, 225
GetSurface, 225
GetWorldAngularAccel, 226
GetWorldAngularVel, 226
GetWorldLinearAccel, 226
GetWorldLinearVel, 226
Init, 226
IsPlaceable, 226
link, 229
Load, 227
placeable, 229
ProcessMsg, 227
SetCategoryBits, 227
SetCollideBits, 227
SetCollision, 227
SetContactsEnabled, 228
SetLaserRetro, 228
SetMaxContacts, 228
SetScale, 228
SetShape, 228
SetState, 229
shape, 229
UpdateParameters, 229
gazebo::physics::CollisionState, 229
~CollisionState, 231
CollisionState, 231
FillSDF, 231
GetPose, 231
IsZero, 232
Load, 232
operator<<, 233
operator+, 232
operator-, 232
operator=, 233
gazebo::physics::Contact, 260
~Contact, 261
collision1, 262
collision2, 262
Contact, 261
count, 262
DebugString, 261
depths, 262
FillMsg, 261
normals, 263
operator=, 262
positions, 263
Reset, 262
time, 263
world, 263
wrench, 263
gazebo::physics::ContactManager, 263
~ContactManager, 264
Clear, 264
ContactManager, 264
CreateFilter, 264, 265
GetContact, 265
GetContactCount, 265
GetContacts, 265
Init, 265
NewContact, 266
PublishContacts, 266
ResetCount, 266
gazebo::physics::ContactPublisher, 266
collisionNames, 267
collisions, 267
contacts, 267
publisher, 267
gazebo::physics::CylinderShape, 275
~CylinderShape, 277
CylinderShape, 277
FillMsg, 277
GetLength, 277
GetRadius, 277
Init, 278
ProcessMsg, 278
SetLength, 278
SetRadius, 278
SetScale, 278
SetSize, 278
gazebo::physics::DARTBallJoint, 279
~DARTBallJoint, 281
DARTBallJoint, 281
dtBallJoint, 284
GetAnchor, 281
GetAngleImpl, 282
GetGlobalAxis, 282
GetMaxForce, 282
GetVelocity, 283

Init, 283
Load, 283
SetForceImpl, 283
SetMaxForce, 283
SetVelocity, 284
gazebo::physics::DARTBoxShape, 284
~DARTBoxShape, 286
DARTBoxShape, 285
SetSize, 286
gazebo::physics::DARTCollision, 286
~DARTCollision, 288
DARTCollision, 288
Fini, 288
GetBoundingBox, 288
GetCategoryBits, 289
GetCollideBits, 289
GetDARTBodyNode, 289
GetDARTCollisionShape, 289
Init, 289
Load, 289
OnPoseChange, 290
SetCategoryBits, 290
SetCollideBits, 290
SetDARTCollisionShape, 290
gazebo::physics::DARTCylinderShape, 290
~DARTCylinderShape, 292
DARTCylinderShape, 292
SetSize, 292
gazebo::physics::DARTHeightmapShape, 292
~DARTHeightmapShape, 294
DARTHeightmapShape, 294
Init, 294
gazebo::physics::DARTHinge2Joint, 294
~DARTHinge2Joint, 296
DARTHinge2Joint, 296
dtUniveralJoint, 299
GetAnchor, 297
GetAngleImpl, 297
GetGlobalAxis, 297
GetMaxForce, 297
GetVelocity, 298
Init, 298
Load, 298
SetAxis, 298
SetForceImpl, 298
SetMaxForce, 299
SetVelocity, 299
gazebo::physics::DARTHingeJoint, 299
~DARTHingeJoint, 301
DARTHingeJoint, 301
dtRevoluteJoint, 304
GetAnchor, 302
GetAngleImpl, 302
GetGlobalAxis, 302
GetMaxForce, 302
GetVelocity, 303
Init, 303
Load, 303
SetAxis, 303
SetForceImpl, 303
SetMaxForce, 304
SetVelocity, 304
gazebo::physics::DARTJoint, 304
~DARTJoint, 307
ApplyDamping, 307
AreConnected, 307
Attach, 307
DARTJoint, 307
dartPhysicsEngine, 313
Detach, 307
dtChildBodyNode, 313
dtJoint, 313
GetAngleCount, 308
GetAttribute, 308
GetDARTJoint, 308
GetDARTModel, 308
GetForce, 308
GetForceTorque, 309
GetHighStop, 309
GetJointLink, 309
GetLinkForce, 310
GetLinkTorque, 310
GetLowStop, 310
Init, 311
Load, 311
Reset, 311
SetAnchor, 311
SetAttribute, 311
SetDamping, 312
SetForce, 312
SetForceImpl, 312
SetHighStop, 313
SetLowStop, 313
gazebo::physics::DARTLink, 313
~DARTLink, 316
AddDARTChildJoint, 316
AddForce, 316
AddForceAtRelativePosition, 317
AddForceAtWorldPosition, 317
AddRelativeForce, 317
AddRelativeTorque, 317
AddTorque, 318
DARTLink, 316
Fini, 318
GetDARTBodyNode, 318
GetDARTModel, 318
GetDARTPhysics, 318
GetDARTWorld, 318

GetEnabled, 318
GetGravityMode, 319
GetKinematic, 319
GetWorldAngularVel, 319
GetWorldCoGLinearVel, 319
GetWorldForce, 319
GetWorldLinearVel, 320
GetWorldTorque, 320
Init, 320
Load, 321
OnPoseChange, 321
SetAngularDamping, 321
SetAngularVel, 321
SetAutoDisable, 321
SetDARTParentJoint, 321
SetEnabled, 322
SetForce, 322
SetGravityMode, 322
SetKinematic, 322
SetLinearDamping, 322
SetLinearVel, 323
SetLinkStatic, 323
SetSelfCollide, 323
SetTorque, 323
updateDirtyPoseFromDARTTransformation, 323
gazebo::physics::DARTMeshShape, 324
~DARTMeshShape, 325
DARTMeshShape, 325
Init, 325
Load, 325
Update, 325
gazebo::physics::DARTModel, 326
~DARTModel, 327
BackupState, 327
DARTModel, 327
dtConfig, 328
dtSkeleton, 328
dtVelocity, 328
Fini, 327
GetDARTPhysics, 328
GetDARTSkeleton, 328
GetDARTWorld, 328
Init, 328
Load, 328
RestoreState, 328
Update, 328
gazebo::physics::DARTMultiRayShape, 328
gazebo::physics::DARTPhysics, 330
~DARTPhysics, 332
CreateCollision, 332
CreateJoint, 332
CreateLink, 332
CreateModel, 333
CreateShape, 333
DARTParam, 332
DARTPhysics, 332
DebugPrint, 333
Fini, 333
GetDARTWorld, 333
GetParam, 333
GetType, 334
Init, 334
InitForThread, 334
Load, 334
MAX_CONTACTS, 332
MIN_STEP_SIZE, 332
OnPhysicsMsg, 334
OnRequest, 335
Reset, 335
SetGravity, 335
SetSeed, 335
UpdateCollision, 335
UpdatePhysics, 335
gazebo::physics::DARTPlaneShape, 336
~DARTPlaneShape, 337
CreatePlane, 337
DARTPlaneShape, 337
SetAltitude, 337
gazebo::physics::DARTRayShape, 338
gazebo::physics::DARTScrewJoint, 339
~DARTScrewJoint, 341
DARTScrewJoint, 341
dartScrewJoint, 344
GetAngleImpl, 342
GetGlobalAxis, 342
GetMaxForce, 342
GetThreadPitch, 342
GetVelocity, 343
Init, 343
Load, 343
SetAxis, 343
SetForceImpl, 343
SetMaxForce, 344
SetThreadPitch, 344
SetVelocity, 344
gazebo::physics::DARTSliderJoint, 345
~DARTSliderJoint, 347
DARTSliderJoint, 347
dtPrismaticJoint, 349
GetAnchor, 347
GetAngleImpl, 347
GetGlobalAxis, 347
GetMaxForce, 348
GetVelocity, 348
Init, 348
Load, 348
SetAxis, 348
SetForceImpl, 349

SetMaxForce, 349
SetVelocity, 349
gazebo::physics::DARTSphereShape, 350
~DARTSphereShape, 351
DARTSphereShape, 351
SetRadius, 351
gazebo::physics::DARTTypes, 351
gazebo::physics::DARTUniversalJoint, 352
~DARTUniversalJoint, 354
DARTUniversalJoint, 354
dtUniveralJoint, 357
GetAnchor, 355
GetAngleImpl, 355
GetGlobalAxis, 355
GetMaxForce, 355
GetVelocity, 356
Init, 356
Load, 356
SetAxis, 356
SetForceImpl, 356
SetMaxForce, 357
SetVelocity, 357
gazebo::physics::Entity, 379
~Entity, 382
animation, 389
animationConnection, 389
animationStartPose, 389
connections, 389
dirtyPose, 389
Entity, 382
Fini, 382
GetBoundingBox, 382
GetChildCollision, 382
GetChildLink, 382
GetCollisionBoundingBox, 383
GetDirtyPose, 383
GetInitialRelativePose, 383
GetNearestEntityBelow, 383
GetParentModel, 383
GetRelativeAngularAccel, 384
GetRelativeAngularVel, 384
GetRelativeLinearAccel, 384
GetRelativeLinearVel, 384
GetRelativePose, 384
GetWorldAngularAccel, 385
GetWorldAngularVel, 385
GetWorldLinearAccel, 385
GetWorldLinearVel, 385
GetWorldPose, 385
IsCanonicalLink, 386
IsStatic, 386
Load, 386
node, 389
OnPoseChange, 386
parentEntity, 389
PlaceOnEntity, 386
PlaceOnNearestEntityBelow, 387
prevAnimationTime, 389
requestPub, 390
Reset, 387
scale, 390
SetAnimation, 387
SetCanonicalLink, 387
SetInitialRelativePose, 387
SetName, 387
SetRelativePose, 388
SetStatic, 388
SetWorldPose, 388
SetWorldTwist, 388
StopAnimation, 389
UpdateParameters, 389
visPub, 390
visualMsg, 390
gazebo::physics::Gripper, 453
~Gripper, 454
GetName, 454
Gripper, 454
Init, 454
IsAttached, 455
Load, 455
node, 455
gazebo::physics::HeightmapShape, 465
~HeightmapShape, 467
FillMsg, 468
flipY, 470
GetHeight, 468
GetImage, 468
GetMaxHeight, 468
GetMinHeight, 468
GetPos, 469
GetSize, 469
GetSubSampling, 469
GetURI, 469
GetVertexCount, 469
HeightmapShape, 467
heights, 470
img, 470
Init, 469
Load, 469
ProcessMsg, 470
SetScale, 470
subSampling, 470
vertSize, 470
gazebo::physics::Hinge2Joint
~Hinge2Joint, 472
GetAngleCount, 472
Hinge2Joint, 472
Load, 472

gazebo::physics::Hinge2Joint< T >, 471
gazebo::physics::HingeJoint
~HingeJoint, 473
GetAngleCount, 474
HingeJoint, 473
Init, 474
Load, 474
gazebo::physics::HingeJoint< T >, 472
gazebo::physics::Inertial, 483
~Inertial, 486
GetCoG, 486
GetIXX, 486
GetIXY, 487
GetIXZ, 487
GetIYY, 487
GetIYZ, 487
GetIZZ, 487
GetInertial, 486
GetMOI, 487, 488
GetMass, 487
GetPose, 488
GetPrincipalMoments, 488
GetProductsofInertia, 488
Inertial, 485, 486
Load, 488
operator<<, 492
operator+, 489
operator+=, 489
operator=, 489
ProcessMsg, 489
Reset, 490
Rotate, 490
SetCoG, 490
SetIXX, 491
SetIXY, 491
SetIXZ, 491
SetIYY, 491
SetIYZ, 492
SetIZZ, 492
SetInertiaMatrix, 491
SetMOI, 492
SetMass, 492
UpdateParameters, 492
gazebo::physics::Joint, 496
~Joint, 500
anchorLink, 515
anchorPos, 515
anchorPose, 515
ApplyDamping, 501
applyDamping, 515
AreConnected, 501
Attach, 501
Attribute, 500
CFM, 500
CacheForceTorque, 501
CheckAndTruncateForce, 501
childLink, 515
ConnectJointUpdate, 501
dampingCoefficient, 515
Detach, 502
DisconnectJointUpdate, 502
ERP, 500
effortLimit, 515
FMAX, 500
FUDGE_FACTOR, 500
FillMsg, 502
GetAnchor, 502
GetAngle, 503
GetAngleCount, 503
GetAngleImpl, 503
GetAttribute, 504
GetChild, 504
GetDamping, 504
GetDampingCoefficient, 504
GetEffortLimit, 504
GetForce, 505
GetForceTorque, 505
GetGlobalAxis, 505
GetHighStop, 506
GetInertiaRatio, 506
GetInitialAnchorPose, 506
GetJointLink, 507
GetLinkForce, 507
GetLinkTorque, 507
GetLocalAxis, 507
GetLowStop, 508
GetLowerLimit, 508
GetMaxForce, 508
GetParent, 509
GetUpperLimit, 509
GetVelocity, 509
GetVelocityLimit, 510
HI_STOP, 500
inertiaRatio, 515
Init, 510
Joint, 500
LO_STOP, 500
Load, 510
lowerLimit, 515
model, 516
parentLink, 516
provideFeedback, 516
Reset, 511
STOP_CFM, 500
STOP_ERP, 500
SUSPENSION_CFM, 500
SUSPENSION_ERP, 500
SetAnchor, 511

SetAngle, 511
SetAttribute, 511
SetAxis, 512
SetDamping, 512
SetEffortLimit, 512
SetForce, 512
SetHighStop, 513
SetLowStop, 513
SetMaxForce, 513
SetModel, 514
SetProvideFeedback, 514
SetState, 514
SetVelocity, 514
Update, 514
UpdateParameters, 514
upperLimit, 516
useCFMDamping, 516
VEL, 500
velocityLimit, 516
wrench, 516
gazebo::physics::JointController, 521
AddJoint, 521
JointController, 521
Reset, 521
SetJointPosition, 522
SetJointPositions, 522
Update, 522
gazebo::physics::JointState, 522
∼JointState, 524
FillSDF, 525
GetAngle, 525
GetAngleCount, 525
GetAngles, 525
IsZero, 525
JointState, 524
Load, 526
operator<<, 527
operator+, 526
operator-, 526
operator=, 526
gazebo::physics::JointWrench, 529
body1Force, 530
body1Torque, 530
body2Force, 531
body2Torque, 531
operator+, 530
operator-, 530
operator=, 530
gazebo::physics::Link, 542
∼Link, 547
AddChildJoint, 547
AddForce, 547
AddForceAtRelativePosition, 547
AddForceAtWorldPosition, 548
AddParentJoint, 548
AddRelativeForce, 548
AddRelativeTorque, 548
AddTorque, 548
angularAccel, 562
AttachStaticModel, 549
attachedModelsOffset, 562
cgVisuals, 562
ConnectEnabled, 549
DetachAllStaticModels, 549
DetachStaticModel, 549
DisconnectEnabled, 549
FillMsg, 550
Fini, 550
GetAngularDamping, 550
GetBoundingBox, 550
GetChildJoints, 550
GetChildJointsLinks, 550
GetCollision, 550, 551
GetCollisions, 551
GetEnabled, 551
GetGravityMode, 551
GetInertial, 551
GetKinematic, 552
GetLinearDamping, 552
GetModel, 552
GetParentJoints, 552
GetParentJointsLinks, 552
GetRelativeAngularAccel, 552
GetRelativeAngularVel, 553
GetRelativeForce, 553
GetRelativeLinearAccel, 553
GetRelativeLinearVel, 553
GetRelativeTorque, 553
GetSelfCollide, 554
GetSensorCount, 554
GetSensorName, 554
GetWorldAngularAccel, 554
GetWorldCoGLinearVel, 555
GetWorldCoGPose, 555
GetWorldForce, 555
GetWorldLinearAccel, 555
GetWorldLinearVel, 555, 556
GetWorldTorque, 556
inertial, 562
Init, 556
linearAccel, 563
Link, 547
Load, 556
OnPoseChange, 557
ProcessMsg, 557
RemoveChild, 557
RemoveChildJoint, 557
RemoveCollision, 557

RemoveParentJoint, 557
Reset, 557
ResetPhysicsStates, 558
SetAngularAccel, 558
SetAngularDamping, 558
SetAngularVel, 558
SetAutoDisable, 558
SetCollideMode, 558
SetEnabled, 559
SetForce, 559
SetGravityMode, 559
SetInertial, 559
SetKinematic, 559
SetLaserRetro, 560
SetLinearAccel, 560
SetLinearDamping, 560
SetLinearVel, 560
SetLinkStatic, 560
SetPublishData, 560
SetScale, 561
SetSelected, 561
SetSelfCollide, 561
SetState, 561
SetTorque, 561
Update, 562
UpdateMass, 562
UpdateParameters, 562
UpdateSurface, 562
visuals, 563
Visuals_M, 547
gazebo::physics::LinkState, 563
~LinkState, 565
FillSDF, 566
GetAcceleration, 566
GetCollisionState, 566
GetCollisionStateCount, 567
GetCollisionStates, 567
GetPose, 567
GetVelocity, 567
GetWrench, 567
IsZero, 567
LinkState, 565
Load, 568
operator<<, 570
operator+, 568
operator-, 568
operator=, 569
SetRealTime, 569
SetSimTime, 569
SetWallTime, 569
gazebo::physics::MapShape, 579
~MapShape, 581
FillMsg, 581
GetGranularity, 581
GetHeight, 581
GetScale, 582
GetThreshold, 582
GetURI, 582
Init, 582
Load, 582
MapShape, 581
ProcessMsg, 583
SetScale, 583
Update, 583
gazebo::physics::MeshShape, 621
~MeshShape, 622
FillMsg, 622
GetMeshURI, 622
GetSize, 623
Init, 623
mesh, 624
MeshShape, 622
ProcessMsg, 623
SetMesh, 623
SetScale, 623
submesh, 624
Update, 624
gazebo::physics::Model, 624
~Model, 628
AttachStaticModel, 628
attachedModels, 638
attachedModelsOffset, 638
DetachStaticModel, 628
FillMsg, 629
Fini, 629
GetAutoDisable, 629
GetBoundingBox, 629
GetGripper, 629
GetGripperCount, 630
GetJoint, 630
GetJointController, 630
GetJointCount, 630
GetJoints, 630
GetLink, 630
GetLinks, 631
GetPluginCount, 631
GetRelativeAngularAccel, 631
GetRelativeAngularVel, 631
GetRelativeLinearAccel, 631
GetRelativeLinearVel, 632
GetSDF, 632
GetSensorCount, 632
GetWorldAngularAccel, 632
GetWorldAngularVel, 632
GetWorldLinearAccel, 633
GetWorldLinearVel, 633
Init, 633
jointPub, 638

Load, 633
LoadJoints, 633
LoadPlugins, 634
Model, 628
OnPoseChange, 634
ProcessMsg, 634
RemoveChild, 634
Reset, 634
SetAngularAccel, 634
SetAngularVel, 634
SetAutoDisable, 635
SetCollideMode, 635
SetEnabled, 635
SetGravityMode, 635
SetJointAnimation, 635
SetJointPosition, 635
SetJointPositions, 636
SetLaserRetro, 636
SetLinearAccel, 636
SetLinearVel, 636
SetLinkWorldPose, 636, 637
SetScale, 637
SetState, 637
StopAnimation, 637
Update, 637
UpdateParameters, 637
gazebo::physics::ModelState, 641
∼ModelState, 644
FillSDF, 644
GetJointState, 644
GetJointStateCount, 645
GetJointStates, 645
GetLinkState, 645
GetLinkStateCount, 646
GetLinkStates, 646
GetPose, 646
HasJointState, 646
HasLinkState, 647
IsZero, 647
Load, 647
ModelState, 643
operator<<, 649
operator+, 648
operator-, 648
operator=, 648
SetRealTime, 648
SetSimTime, 648
SetWallTime, 649
gazebo::physics::MultiRayShape, 664
∼MultiRayShape, 667
AddRay, 667
ConnectNewLaserScans, 667
DisconnectNewLaserScans, 668
FillMsg, 668
GetFiducial, 668
GetMaxAngle, 668
GetMaxRange, 669
GetMinAngle, 669
GetMinRange, 669
GetRange, 669
GetResRange, 669
GetRetro, 669
GetSampleCount, 670
GetScanResolution, 670
GetVerticalMaxAngle, 670
GetVerticalMinAngle, 670
GetVerticalSampleCount, 670
GetVerticalScanResolution, 670
horzElem, 671
Init, 671
MultiRayShape, 667
newLaserScans, 672
offset, 672
ProcessMsg, 671
rangeElem, 672
rayElem, 672
rays, 672
scanElem, 672
SetScale, 671
Update, 671
UpdateRays, 671
vertElem, 672
gazebo::physics::PhysicsEngine, 706
∼PhysicsEngine, 710
contactManager, 719
CreateCollision, 710
CreateJoint, 711
CreateLink, 711
CreateModel, 711
CreateShape, 711
DebugPrint, 711
Fini, 712
GetAutoDisableFlag, 712
GetContactManager, 712
GetContactMaxCorrectingVel, 712
GetContactSurfaceLayer, 712
GetGravity, 712
GetMaxContacts, 713
GetMaxStepSize, 713
GetParam, 713
GetPhysicsUpdateMutex, 713
GetRealTimeUpdateRate, 713
GetSORPGSIters, 714
GetSORPGSPreconIters, 714
GetSORPGSW, 714
GetTargetRealTimeFactor, 714
GetType, 714
GetUpdatePeriod, 714

GetWorldCFM, 715
GetWorldERP, 715
Init, 715
InitForThread, 715
Load, 715
maxStepSize, 719
node, 719
OnPhysicsMsg, 715
OnRequest, 716
PhysicsEngine, 710
physicsSub, 719
physicsUpdateMutex, 720
realTimeUpdateRate, 720
requestSub, 720
Reset, 716
responsePub, 720
sdf, 720
SetAutoDisableFlag, 716
SetContactMaxCorrectingVel, 716
SetContactSurfaceLayer, 716
SetGravity, 717
SetMaxContacts, 717
SetMaxStepSize, 717
SetParam, 717
SetRealTimeUpdateRate, 717
SetSORPGSIters, 718
SetSORPGSPreconIters, 718
SetSORPGSW, 718
SetSeed, 718
SetTargetRealTimeFactor, 718
SetWorldCFM, 719
SetWorldERP, 719
targetRealTimeFactor, 720
UpdateCollision, 719
UpdatePhysics, 719
world, 720
gazebo::physics::PhysicsFactory, 720
IsRegistered, 721
NewPhysicsEngine, 721
RegisterAll, 721
RegisterPhysicsEngine, 721
gazebo::physics::PlaneShape, 728
∼PlaneShape, 730
CreatePlane, 730
FillMsg, 730
GetNormal, 731
GetSize, 731
Init, 731
PlaneShape, 730
ProcessMsg, 731
SetAltitude, 731
SetNormal, 731
SetScale, 732
SetSize, 732
gazebo::physics::RayShape, 786
∼RayShape, 788
contactFiducial, 791
contactLen, 791
contactRetro, 791
FillMsg, 788
GetFiducial, 788
GetGlobalPoints, 788
GetIntersection, 789
GetLength, 789
GetRelativePoints, 789
GetRetro, 789
globalEndPos, 791
globalStartPos, 791
Init, 789
ProcessMsg, 789
RayShape, 788
relativeEndPos, 791
relativeStartPos, 791
SetFiducial, 790
SetLength, 790
SetPoints, 790
SetRetro, 790
SetScale, 790
Update, 790
gazebo::physics::Road, 804
∼Road, 805
Init, 805
Load, 805
Road, 805
gazebo::physics::ScrewJoint
∼ScrewJoint, 833
fakeAnchor, 835
GetAnchor, 833
GetAngleCount, 834
GetThreadPitch, 834
Load, 834
ScrewJoint, 833
SetAnchor, 834
SetThreadPitch, 835
threadPitch, 835
gazebo::physics::ScrewJoint< T >, 832
gazebo::physics::Shape, 861
∼Shape, 863
collisionParent, 864
FillMsg, 863
GetScale, 863
Init, 864
ProcessMsg, 864
scale, 865
SetScale, 864
Shape, 863
gazebo::physics::SimbodyBallJoint, 865
∼SimbodyBallJoint, 867

GetAnchor, 868
GetAngleImpl, 868
GetAxis, 868
GetGlobalAxis, 868
GetMaxForce, 868
GetVelocity, 869
Init, 869
Load, 869
SetDamping, 869
SetForceImpl, 869
SetHighStop, 870
SetLowStop, 870
SetMaxForce, 870
SetVelocity, 870
SimbodyBallJoint, 867
gazebo::physics::SimbodyBoxShape, 871
~SimbodyBoxShape, 872
SetSize, 872
SimbodyBoxShape, 872
gazebo::physics::SimbodyCollision, 872
~SimbodyCollision, 874
GetBoundingBox, 874
GetCollisionShape, 874
Load, 874
OnPoseChange, 875
SetCategoryBits, 875
SetCollideBits, 875
SetCollisionShape, 875
SimbodyCollision, 874
gazebo::physics::SimbodyCylinderShape, 875
~SimbodyCylinderShape, 877
SetSize, 877
SimbodyCylinderShape, 877
gazebo::physics::SimbodyHeightmapShape, 877
~SimbodyHeightmapShape, 879
Init, 879
SimbodyHeightmapShape, 879
gazebo::physics::SimbodyHinge2Joint, 879
~SimbodyHinge2Joint, 881
GetAnchor, 882
GetAngleImpl, 882
GetAxis, 882
GetGlobalAxis, 882
GetHighStop, 882
GetLowStop, 883
GetMaxForce, 883
GetVelocity, 883
Init, 884
Load, 884
SetAxis, 884
SetDamping, 884
SetForceImpl, 884
SetHighStop, 884
SetLowStop, 885
SetMaxForce, 885
SetVelocity, 885
SimbodyHinge2Joint, 881
gazebo::physics::SimbodyHingeJoint, 886
~SimbodyHingeJoint, 887
GetAngleImpl, 888
GetGlobalAxis, 888
GetHighStop, 888
GetLowStop, 888
GetMaxForce, 889
GetVelocity, 889
Load, 889
RestoreSimbodyState, 890
SaveSimbodyState, 890
SetAxis, 890
SetDamping, 890
SetForceImpl, 890
SetHighStop, 891
SetLowStop, 891
SetMaxForce, 891
SetVelocity, 891
SimbodyHingeJoint, 887
gazebo::physics::SimbodyJoint, 892
~SimbodyJoint, 894
AreConnected, 894
CacheForceTorque, 894
constraint, 899
damper, 899
defxAB, 899
Detach, 894
GetAnchor, 894
GetAttribute, 895
GetForce, 895
GetForceTorque, 895
GetJointLink, 896
GetLinkForce, 896
GetLinkTorque, 896
isReversed, 899
limitForce, 899
Load, 897
mobod, 900
mustBreakLoopHere, 900
physicsInitialized, 900
Reset, 897
RestoreSimbodyState, 897
SaveSimbodyState, 897
SetAnchor, 897
SetAttribute, 897, 898
SetAxis, 898
SetDamping, 898
SetForce, 898
SetForceImpl, 899
SimbodyJoint, 894
simbodyPhysics, 900

world, 900
xCB, 900
xPA, 900
gazebo::physics::SimbodyLink, 900
~SimbodyLink, 903
AddForce, 903
AddForceAtRelativePosition, 903
AddForceAtWorldPosition, 904
AddRelativeForce, 904
AddRelativeTorque, 904
AddTorque, 904
Fini, 905
GetEffectiveMassProps, 905
GetEnabled, 905
GetGravityMode, 905
GetMassProperties, 905
GetWorldAngularVel, 905
GetWorldCoGLinearVel, 905
GetWorldForce, 905
GetWorldLinearVel, 906
GetWorldTorque, 906
Init, 906
Load, 907
masterMobod, 909
mustBeBaseLink, 909
OnPoseChange, 907
physicsInitialized, 909
RestoreSimbodyState, 907
SaveSimbodyState, 907
SetAngularDamping, 907
SetAngularVel, 907
SetAutoDisable, 907
SetDirtyPose, 908
SetEnabled, 908
SetForce, 908
SetGravityMode, 908
SetLinearDamping, 908
SetLinearVel, 908
SetLinkStatic, 909
SetSelfCollide, 909
SetTorque, 909
SimbodyLink, 903
slaveMobods, 909
slaveWelds, 909
gazebo::physics::SimbodyMeshShape, 910
~SimbodyMeshShape, 911
Init, 911
Load, 911
SimbodyMeshShape, 911
gazebo::physics::SimbodyModel, 912
~SimbodyModel, 913
Init, 913
Load, 913
SimbodyModel, 913
gazebo::physics::SimbodyMultiRayShape, 913
~SimbodyMultiRayShape, 915
AddRay, 915
SimbodyMultiRayShape, 915
UpdateRays, 915
gazebo::physics::SimbodyPhysics, 915
~SimbodyPhysics, 918
contact, 924
CreateCollision, 918
CreateJoint, 918
CreateLink, 918
CreateModel, 919
CreateShape, 919
DebugPrint, 919
discreteForces, 924
Fini, 919
forces, 924
GetDynamicsWorld, 919
GetPose, 919
GetType, 920
GetTypeString, 920
gravity, 924
Init, 920
InitForThread, 920
InitModel, 920
integ, 924
Load, 921
matter, 924
OnPhysicsMsg, 921
OnRequest, 921
Pose2Transform, 921
QuadToQuad, 922
Reset, 922
SetGravity, 922
SetSeed, 922
SimbodyPhysics, 918
simbodyPhysicsInitialized, 924
simbodyPhysicsStepped, 924
system, 924
tracker, 924
Transform2Pose, 923
UpdateCollision, 923
UpdatePhysics, 923
Vec3ToVector3, 923
Vector3ToVec3, 923
gazebo::physics::SimbodyPlaneShape, 924
~SimbodyPlaneShape, 926
CreatePlane, 926
SetAltitude, 926
SimbodyPlaneShape, 926
gazebo::physics::SimbodyRayShape, 926
~SimbodyRayShape, 928
GetIntersection, 928
SetPoints, 928

SimbodyRayShape, 928
Update, 929
gazebo::physics::SimbodyScrewJoint, 929
~SimbodyScrewJoint, 932
GetAngleImpl, 932
GetGlobalAxis, 932
GetHighStop, 932
GetLowStop, 932
GetMaxForce, 933
GetThreadPitch, 933
GetVelocity, 933
Init, 934
Load, 934
SetAxis, 934
SetDamping, 934
SetForceImpl, 934
SetHighStop, 935
SetLowStop, 935
SetMaxForce, 935
SetThreadPitch, 935
SetVelocity, 936
SimbodyScrewJoint, 931
gazebo::physics::SimbodySliderJoint, 936
~SimbodySliderJoint, 938
GetAngleImpl, 939
GetGlobalAxis, 939
GetHighStop, 939
GetLowStop, 939
GetMaxForce, 940
GetVelocity, 940
Load, 940
SetAxis, 940
SetDamping, 941
SetForceImpl, 941
SetHighStop, 941
SetLowStop, 941
SetMaxForce, 942
SetVelocity, 942
SimbodySliderJoint, 938
gazebo::physics::SimbodySphereShape, 942
~SimbodySphereShape, 944
SetRadius, 944
SimbodySphereShape, 943
gazebo::physics::SimbodyUniversalJoint, 944
~SimbodyUniversalJoint, 947
GetAnchor, 947
GetAngleImpl, 947
GetAxis, 947
GetGlobalAxis, 947
GetHighStop, 947
GetLowStop, 948
GetMaxForce, 948
GetVelocity, 948
Init, 949
Load, 949
SetAxis, 949
SetDamping, 949
SetForceImpl, 949
SetHighStop, 950
SetLowStop, 950
SetMaxForce, 950
SetVelocity, 950
SimbodyUniversalJoint, 946
gazebo::physics::SliderJoint
~SliderJoint, 974
fakeAnchor, 975
GetAnchor, 974
GetAngleCount, 975
Load, 975
SetAnchor, 975
SliderJoint, 974
gazebo::physics::SliderJoint< T >, 973
gazebo::physics::SphereShape, 986
~SphereShape, 987
FillMsg, 987
GetRadius, 987
Init, 988
ProcessMsg, 988
SetRadius, 988
SetScale, 988
SphereShape, 987
gazebo::physics::State, 998
~State, 999
GetName, 1000
GetRealTime, 1000
GetSimTime, 1000
GetWallTime, 1000
Load, 1000
name, 1002
operator-, 1000
operator=, 1001
realTime, 1002
SetName, 1001
SetRealTime, 1001
SetSimTime, 1001
SetWallTime, 1002
simTime, 1002
State, 999
wallTime, 1002
gazebo::physics::SurfaceParams, 1020
~SurfaceParams, 1021
bounce, 1022
bounceThreshold, 1022
cfm, 1022
collideWithoutContact, 1022
collideWithoutContactBitmask, 1022
erp, 1023
fdir1, 1023

FillMsg, 1022
kd, 1023
kp, 1023
Load, 1022
maxVel, 1023
minDepth, 1023
mu1, 1024
mu2, 1024
ProcessMsg, 1022
slip1, 1024
slip2, 1024
SurfaceParams, 1021
gazebo::physics::TrajectoryInfo, 1061
duration, 1061
endTime, 1061
id, 1061
startTime, 1061
translated, 1061
type, 1061
gazebo::physics::UniversalJoint
∼UniversalJoint, 1064
GetAngleCount, 1065
Load, 1065
UniversalJoint, 1064
gazebo::physics::UniversalJoint< T >, 1063
gazebo::physics::World, 1157
∼World, 1160
Clear, 1160
ClearModels, 1160
dirtyPoses, 1168
DisableAllModels, 1160
EnableAllModels, 1160
EnablePhysicsEngine, 1160
Fini, 1161
GetByName, 1161
GetEnablePhysicsEngine, 1161
GetEntity, 1161
GetEntityBelowPoint, 1161
GetModel, 1162
GetModelBelowPoint, 1162
GetModelCount, 1163
GetModels, 1163
GetName, 1163
GetPauseTime, 1163
GetPhysicsEngine, 1163
GetRealTime, 1163
GetRunning, 1164
GetSelectedEntity, 1164
GetSetWorldPoseMutex, 1164
GetSimTime, 1164
GetSphericalCoordinates, 1164
GetStartTime, 1164
Init, 1165
InsertModelFile, 1165
InsertModelSDF, 1165
InsertModelString, 1165
IsLoaded, 1165
IsPaused, 1165
Load, 1166
LoadPlugin, 1166
PrintEntityTree, 1166
PublishModelPose, 1166
RemovePlugin, 1166
Reset, 1167
ResetEntities, 1167
ResetTime, 1167
Run, 1167
Save, 1167
SetPaused, 1167
SetSimTime, 1167
SetState, 1168
StepWorld, 1168
Stop, 1168
StripWorldName, 1168
UpdateStateSDF, 1168
World, 1160
gazebo::physics::WorldState, 1170
∼WorldState, 1173
FillSDF, 1173
GetModelState, 1173
GetModelStateCount, 1173
GetModelStates, 1173, 1174
HasModelState, 1174
IsZero, 1174
Load, 1174
operator<<, 1176
operator+, 1175
operator-, 1175
operator=, 1175
SetRealTime, 1175
SetSimTime, 1176
SetWallTime, 1176
SetWorld, 1176
WorldState, 1172
gazebo::rendering, 118
ArrowVisualPtr, 121
AxisVisualPtr, 121
COMVisualPtr, 121
CameraPtr, 121
CameraVisualPtr, 121
ContactVisualPtr, 121
DepthCameraPtr, 121
DynamicLinesPtr, 121
GpuLaserPtr, 121
JointVisualPtr, 121
LaserVisualPtr, 121
LightPtr, 121
RENDERING_LINE_LIST, 122

RENDERING_LINE_STRIP, 122
RENDERING_MESH_RESOURCE, 122
RENDERING_POINT_LIST, 122
RENDERING_TRIANGLE_FAN, 122
RENDERING_TRIANGLE_LIST, 122
RENDERING_TRIANGLE_STRIP, 122
RFIDTagVisualPtr, 121
RFIDVisualPtr, 121
RenderOpType, 122
ScenePtr, 121
SelectionObjPtr, 122
SonarVisualPtr, 122
UserCameraPtr, 122
VisualPtr, 122
WindowManagerPtr, 122
WrenchVisualPtr, 122
gazebo::rendering::ArrowVisual, 149
~ArrowVisual, 151
ArrowVisual, 150
Load, 151
ShowRotation, 151
gazebo::rendering::AxisVisual, 155
~AxisVisual, 156
AxisVisual, 156
Load, 156
ScaleXAxis, 156
ScaleYAxis, 156
ScaleZAxis, 156
SetAxisMaterial, 157
ShowRotation, 157
gazebo::rendering::COMVisual, 244
~COMVisual, 246
COMVisual, 245
Load, 246
gazebo::rendering::Camera, 186
~Camera, 192
animState, 210
AnimationComplete, 193
AttachToVisual, 193
AttachToVisualImpl, 193, 194
bayerFrameBuffer, 210
Camera, 192
camera, 210
captureData, 210
captureDataOnce, 210
ConnectNewImageFrame, 194
connections, 211
CreateRenderTexture, 195
DisconnectNewImageFrame, 195
EnableSaveFrame, 195
Fini, 195
GetAspectRatio, 195
GetAvgFPS, 195
GetCameraToViewportRay, 196
GetCaptureData, 196
GetDirection, 196
GetFarClip, 196
GetFrameFilename, 196
GetHFOV, 196
GetImageByteSize, 197
GetImageData, 197
GetImageDepth, 197
GetImageFormat, 198
GetImageHeight, 198
GetImageWidth, 198
GetInitialized, 198
GetLastRenderWallTime, 198
GetName, 198
GetNearClip, 199
GetOgreCamera, 199
GetPitchNode, 199
GetRenderRate, 199
GetRenderTexture, 199
GetRight, 199
GetScene, 200
GetSceneNode, 200
GetScreenshotPath, 200
GetTextureHeight, 200
GetTextureWidth, 200
GetTriangleCount, 200
GetUp, 201
GetVFOV, 201
GetViewport, 201
GetViewportHeight, 201
GetViewportWidth, 201
GetWindowId, 201
GetWorldPointOnPlane, 202
GetWorldPose, 202
GetWorldPosition, 202
GetWorldRotation, 202
GetZValue, 202
imageFormat, 211
imageHeight, 211
imageWidth, 211
Init, 203
initialized, 211
IsAnimating, 203
IsVisible, 203
lastRenderWallTime, 211
Load, 203, 204
MoveToPosition, 204
MoveToPositions, 204
name, 211
newData, 211
newImageFrame, 211
onAnimationComplete, 211
pitchNode, 211
PostRender, 204

prevAnimTime, 212
ReadPixelBuffer, 205
Render, 205
RenderImpl, 205
renderTarget, 212
renderTexture, 212
requests, 212
RotatePitch, 205
RotateYaw, 205
saveCount, 212
SaveFrame, 205, 206
saveFrameBuffer, 212
scene, 212
sceneNode, 212
screenshotPath, 212
sdf, 212
SetAspectRatio, 206
SetCaptureData, 206
SetCaptureDataOnce, 206
SetClipDist, 206
SetHFOV, 207
SetImageHeight, 207
SetImageSize, 207
SetImageWidth, 207
SetName, 207
SetRenderRate, 207
SetRenderTarget, 208
SetSaveFramePathname, 208
SetScene, 208
SetSceneNode, 208
SetWindowId, 208
SetWorldPose, 208
SetWorldPosition, 209
SetWorldRotation, 209
ShowWireframe, 209
textureHeight, 212
textureWidth, 212
ToggleShowWireframe, 209
TrackVisual, 209
TrackVisualImpl, 209, 210
Translate, 210
Update, 210
viewport, 213
windowId, 213
gazebo::rendering::CameraVisual, 217
∼CameraVisual, 218
CameraVisual, 218
Load, 218
gazebo::rendering::ContactVisual, 272
∼ContactVisual, 273
ContactVisual, 273
SetEnabled, 273
gazebo::rendering::Conversions, 273
Convert, 274, 275
gazebo::rendering::DepthCamera, 357
∼DepthCamera, 359
ConnectNewDepthFrame, 359
ConnectNewRGBPointCloud, 360
CreateDepthTexture, 360
DepthCamera, 359
depthTarget, 362
depthTexture, 362
depthViewport, 362
DisconnectNewDepthFrame, 360
DisconnectNewRGBPointCloud, 360
Fini, 361
GetDepthData, 361
Init, 361
Load, 361
PostRender, 361
SetDepthTarget, 361
gazebo::rendering::DummyPageProvider, 370
loadProceduralPage, 370
prepareProceduralPage, 370
unloadProceduralPage, 371
unprepareProceduralPage, 371
gazebo::rendering::DynamicLines, 371
∼DynamicLines, 372
AddPoint, 373
Clear, 373
DynamicLines, 372
GetMovableType, 373
getMovableType, 373
GetPoint, 373
GetPointCount, 374
SetColor, 374
SetPoint, 374
Update, 374
gazebo::rendering::DynamicRenderable, 375
∼DynamicRenderable, 376
CreateVertexDeclaration, 376
DynamicRenderable, 376
FillHardwareBuffers, 376
getBoundingRadius, 377
GetMovableType, 377
GetOperationType, 377
getSquaredViewDepth, 377
indexBufferCapacity, 378
Init, 377
PrepareHardwareBuffers, 378
SetOperationType, 378
vertexBufferCapacity, 378
gazebo::rendering::Events, 405
ConnectCreateScene, 406
ConnectRemoveScene, 406
createScene, 407
DisconnectCreateScene, 406
DisconnectRemoveScene, 407

removeScene, 407
gazebo::rendering::FPSViewController, 422
~FPSViewController, 424
FPSViewController, 424
GetTypeString, 424
HandleKeyPressEvent, 424
HandleKeyReleaseEvent, 424
HandleMouseEvent, 424
Init, 425
Update, 425
gazebo::rendering::GUIOverlay, 455
~GUIOverlay, 456
AttachCameraToImage, 456, 457
ButtonCallback, 457
CreateWindow, 457
GUIOverlay, 456
HandleKeyPressEvent, 457
HandleKeyReleaseEvent, 458
HandleMouseEvent, 458
Hide, 458
Init, 458
IsInitialized, 458
LoadLayout, 459
Resize, 459
Show, 459
Update, 459
gazebo::rendering::GpuLaser, 429
~GpuLaser, 432
cameraCount, 437
chfov, 437
ConnectNewLaserFrame, 432
CreateLaserTexture, 432
cvfov, 437
DisconnectNewLaserFrame, 432
far, 437
Fini, 433
GetCameraCount, 433
GetCosHorzFOV, 433
GetCosVertFOV, 433
GetFarClip, 433
GetHorzFOV, 433
GetHorzHalfAngle, 433
GetLaserData, 434
GetNearClip, 434
GetRayCountRatio, 434
GetVertFOV, 434
GetVertHalfAngle, 434
GpuLaser, 432
hfov, 438
horzHalfAngle, 438
Init, 434
IsHorizontal, 435
isHorizontal, 438
Load, 435
near, 438
notifyRenderSingleObject, 435
PostRender, 435
rayCountRatio, 438
SetCameraCount, 435
SetCosHorzFOV, 435
SetCosVertFOV, 435
SetFarClip, 436
SetHorzFOV, 436
SetHorzHalfAngle, 436
SetIsHorizontal, 436
SetNearClip, 436
SetRangeCount, 436
SetRayCountRatio, 437
SetVertFOV, 437
SetVertHalfAngle, 437
vertHalfAngle, 438
vfov, 438
gazebo::rendering::Grid, 450
~Grid, 451
Enable, 451
GetCellCount, 451
GetCellLength, 451
GetColor, 452
GetHeight, 452
GetLineWidth, 452
GetSceneNode, 452
Grid, 451
Init, 452
SetCellCount, 452
SetCellLength, 452
SetColor, 453
SetHeight, 453
SetLineWidth, 453
SetUserData, 453
gazebo::rendering::GzTerrainMatGen, 459
~GzTerrainMatGen, 460
GzTerrainMatGen, 460
gazebo::rendering::GzTerrainMatGen::SM2Profile, 975
~SM2Profile, 977
addTechnique, 977
generate, 977
generateForCompositeMap, 977
SM2Profile, 977
UpdateParams, 977
UpdateParamsForCompositeMap, 977
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperCg, 857
defaultVpParams, 858
generateFragmentProgram, 858
generateVertexProgram, 858
generateVertexProgramSource, 858
generateVpDynamicShadows, 858
generateVpDynamicShadowsParams, 858

generateVpFooter, 858
generateVpHeader, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 859
defaultVpParams, 860
generateFpDynamicShadows, 860
generateFpDynamicShadowsHelpers, 860
generateFpDynamicShadowsParams, 860
generateFpFooter, 860
generateFpHeader, 860
generateFpLayer, 860
generateFragmentProgram, 860
generateFragmentProgramSource, 861
generateVertexProgram, 861
generateVertexProgramSource, 861
generateVpDynamicShadows, 861
generateVpDynamicShadowsParams, 861
generateVpFooter, 861
generateVpHeader, 861
updateParams, 861
updateVpParams, 861
gazebo::rendering::Heightmap, 460
∼Heightmap, 462
Flatten, 462
GetAvgHeight, 462
GetHeight, 462
GetImage, 463
GetMouseHit, 463
GetOgreTerrain, 463
GetTerrainSubdivisionCount, 463
Heightmap, 462
Load, 463
LoadFromMsg, 463
Lower, 464
NumTerrainSubdivisions, 465
Raise, 464
SetWireframe, 464
Smooth, 464
SplitHeights, 465
gazebo::rendering::JointVisual, 527
∼JointVisual, 529
JointVisual, 528
Load, 529
gazebo::rendering::LaserVisual, 534
∼LaserVisual, 535
LaserVisual, 535
SetEmissive, 535
gazebo::rendering::Light, 535
∼Light, 537
FillMsg, 537
GetDiffuseColor, 537
GetDirection, 537
GetName, 538
GetPosition, 538
GetSpecularColor, 538
GetType, 538
GetVisible, 538
Light, 537
Load, 538, 539
LoadFromMsg, 539
OnPoseChange, 539
SetAttenuation, 539
SetCastShadows, 539
SetDiffuseColor, 539
SetDirection, 539
SetLightType, 540
SetName, 540
SetPosition, 540
SetRange, 540
SetSelected, 540
SetSpecularColor, 540
SetSpotFalloff, 541
SetSpotInnerAngle, 541
SetSpotOuterAngle, 541
ShowVisual, 541
ToggleShowVisual, 541
UpdateFromMsg, 541
gazebo::rendering::MovableText, 652
∼MovableText, 655
_setupGeometry, 655
_updateColors, 655
GetAABB, 655
GetBaseline, 655
getBoundingRadius, 655
GetCharHeight, 655
GetColor, 655
GetFont, 656
getLights, 656
getMaterial, 656
getRenderOperation, 656
GetShowOnTop, 656
GetSpaceWidth, 656
getSquaredViewDepth, 656
GetText, 656
getWorldTransforms, 656
H_CENTER, 654
H_LEFT, 654
HorizAlign, 654
Load, 657
MovableText, 655
SetBaseline, 657
SetCharHeight, 657
SetColor, 657
SetFontName, 657
SetShowOnTop, 657
SetSpaceWidth, 658
SetText, 658
SetTextAlignment, 658

Update, 658
V_ABOVE, 655
V_BELOW, 655
VertAlign, 654
visitRenderables, 658
gazebo::rendering::OrbitViewController, 703
~OrbitViewController, 704
GetFocalPoint, 704
GetTypeString, 704
HandleKeyPressEvent, 705
HandleKeyReleaseEvent, 705
HandleMouseEvent, 705
Init, 705
OrbitViewController, 704
SetDistance, 706
SetFocalPoint, 706
Update, 706
gazebo::rendering::Projector, 748
~Projector, 749
GetParent, 749
Load, 749
Projector, 749
SetEnabled, 750
SetTexture, 750
Toggle, 750
gazebo::rendering::RFIDTagVisual, 801
~RFIDTagVisual, 802
RFIDTagVisual, 802
gazebo::rendering::RFIDVisual, 802
~RFIDVisual, 803
RFIDVisual, 803
gazebo::rendering::RTShaderSystem, 810
AddScene, 812
ApplyShadows, 812
AttachEntity, 812
AttachViewport, 812
Clear, 813
DetachEntity, 813
DetachViewport, 813
Fini, 813
GenerateShaders, 813
GetPSSMShadowCameraSetup, 813
Init, 813
LightingModel, 812
RemoveScene, 813
RemoveShadows, 814
SSLM_NormalMapLightingObjectSpace, 812
SSLM_NormalMapLightingTangentSpace, 812
SSLM_PerPixelLighting, 812
SSLM_PerVertexLighting, 812
SetPerPixelLighting, 814
UpdateShaders, 814
gazebo::rendering::RenderEngine, 791
AddResourcePath, 793
CreateScene, 793
DEFERRED, 793
dummyContext, 795
dummyDisplay, 795
dummyWindowId, 795
FORWARD, 793
Fini, 794
GetRenderPathType, 794
GetScene, 794
GetSceneCount, 794
GetWindowManager, 795
Init, 795
Load, 795
NONE, 793
RENDER_PATH_COUNT, 793
RemoveScene, 795
RenderPathType, 793
root, 795
VERTEX, 793
gazebo::rendering::Road2d, 806
~Road2d, 806
Load, 806
Road2d, 806
gazebo::rendering::Scene, 814
~Scene, 819
AddVisual, 819
Clear, 819
CloneVisual, 819
CreateCamera, 819
CreateDepthCamera, 819
CreateGpuLaser, 820
CreateGrid, 820
CreateUserCamera, 820
DrawLine, 821
GZ_SKYX_ALL, 818
GZ_SKYX_CLOUDS, 818
GZ_SKYX_MOON, 818
GZ_SKYX_NONE, 818
GetAmbientColor, 821
GetBackgroundColor, 821
GetCamera, 821
GetCameraCount, 822
GetFirstContact, 822
GetGrid, 822
GetGridCount, 822
GetHeightBelowPoint, 822
GetHeightmap, 823
GetId, 823
GetIdString, 823
GetInitialized, 823
GetLight, 823, 824
GetLightCount, 824
GetManager, 824
GetModelVisualAt, 824

GetName, 824
GetSelectedVisual, 825
GetShadowsEnabled, 825
GetShowClouds, 825
GetSimTime, 825
GetUserCamera, 825
GetUserCameraCount, 826
GetVisual, 826
GetVisualAt, 826, 827
GetVisualBelow, 827
GetVisualCount, 827
GetVisualsBelowPoint, 827
GetWorldVisual, 827
Init, 828
Load, 828
PreRender, 828
PrintSceneGraph, 828
RemoveCamera, 828
RemoveProjectors, 828
RemoveVisual, 828
Scene, 819
SelectVisual, 829
SetAmbientColor, 829
SetBackgroundColor, 829
SetFog, 829
SetGrid, 829
SetShadowsEnabled, 830
SetSkyXMode, 830
SetTransparent, 830
SetVisible, 830
SetWireframe, 830
ShowCOMs, 831
ShowClouds, 830
ShowCollisions, 831
ShowContacts, 831
ShowJoints, 831
SkyXMode, 818
skyx, 832
SnapVisualToNearestBelow, 831
StripSceneName, 831
gazebo::rendering::SelectionObj, 835
gazebo::rendering::SonarVisual, 982
∼SonarVisual, 983
Load, 983
SonarVisual, 983
gazebo::rendering::TransmitterVisual, 1062
∼TransmitterVisual, 1063
Load, 1063
TransmitterVisual, 1063
Update, 1063
gazebo::rendering::UserCamera, 1066
∼UserCamera, 1068
AnimationComplete, 1068
AttachToVisualImpl, 1068
EnableViewController, 1069
Fini, 1069
GetAvgFPS, 1069
GetGUIOverlay, 1069
GetImageHeight, 1070
GetImageWidth, 1070
GetTriangleCount, 1070
GetViewControllerTypeString, 1070
GetVisual, 1070, 1071
HandleKeyPressEvent, 1071
HandleKeyReleaseEvent, 1071
HandleMouseEvent, 1071
Init, 1071
Load, 1071, 1072
MoveToPosition, 1072
MoveToVisual, 1072
PostRender, 1072
Resize, 1072
SetFocalPoint, 1073
SetRenderTarget, 1073
SetViewController, 1073
SetViewportDimensions, 1073
SetWorldPose, 1074
TrackVisualImpl, 1074
Update, 1074
UserCamera, 1068
gazebo::rendering::VideoVisual, 1116
∼VideoVisual, 1117
VideoVisual, 1117
gazebo::rendering::ViewController, 1118
∼ViewController, 1119
camera, 1121
enabled, 1121
GetTypeString, 1119
HandleKeyPressEvent, 1119
HandleKeyReleaseEvent, 1120
HandleMouseEvent, 1120
Init, 1120
SetEnabled, 1120
typeString, 1121
Update, 1121
ViewController, 1119
gazebo::rendering::Visual, 1121
∼Visual, 1127
AttachAxes, 1127
AttachLineVertex, 1127
AttachMesh, 1128
AttachObject, 1128
AttachVisual, 1128
ClearParent, 1128
Clone, 1128
CreateDynamicLine, 1128
DeleteDynamicLine, 1129
DetachObjects, 1129

DetachVisual, 1129
DisableTrackVisual, 1129
EnableTrackVisual, 1129
Fini, 1130
GetAttachedObjectCount, 1130
GetBoundingBox, 1130
GetChild, 1130
GetChildCount, 1130
GetId, 1130
GetMaterialName, 1130
GetMeshName, 1131
GetName, 1131
GetNormalMap, 1131
GetParent, 1131
GetPose, 1131
GetPosition, 1131
GetRootVisual, 1132
GetRotation, 1132
GetScale, 1132
GetScene, 1132
GetSceneNode, 1132
GetShaderType, 1132
GetSubMeshName, 1133
GetTransparency, 1133
GetVisibilityFlags, 1133
GetVisible, 1133
GetWorldPose, 1133
HasAttachedObject, 1134
Init, 1134
InsertMesh, 1134
IsPlane, 1134
IsStatic, 1135
Load, 1135
LoadFromMsg, 1135
LoadPlugin, 1135
MakeStatic, 1135
MoveToPosition, 1136
MoveToPositions, 1136
parent, 1142
RemovePlugin, 1136
scene, 1142
sceneNode, 1142
SetAmbient, 1136
SetCastShadows, 1136
SetDiffuse, 1136
SetEmissive, 1137
SetHighlighted, 1137
SetId, 1137
SetMaterial, 1137
SetName, 1137
SetNormalMap, 1137
SetPose, 1138
SetPosition, 1138
SetRibbonTrail, 1138
SetRotation, 1138
SetScale, 1138
SetScene, 1139
SetShaderType, 1139
SetSkeletonPose, 1139
SetSpecular, 1139
SetTransparency, 1139
SetVisibilityFlags, 1139
SetVisible, 1140
SetWireframe, 1140
SetWorldPose, 1140
SetWorldPosition, 1140
SetWorldRotation, 1140
ShowBoundingBox, 1141
ShowCOM, 1141
ShowCollision, 1141
ShowJoints, 1141
ShowSkeleton, 1141
ToggleVisible, 1141
Update, 1141
UpdateFromMsg, 1141
Visual, 1127
gazebo::rendering::WindowManager, 1144
~WindowManager, 1144
CreateWindow, 1145
Fini, 1145
GetAvgFPS, 1145
GetTriangleCount, 1145
GetWindow, 1145
Moved, 1146
Resize, 1146
SetCamera, 1146
WindowManager, 1144
gazebo::rendering::WireBox, 1146
~WireBox, 1147
Init, 1147
SetVisible, 1147
WireBox, 1147
gazebo::rendering::WrenchVisual, 1177
~WrenchVisual, 1178
Load, 1178
SetEnabled, 1178
WrenchVisual, 1177
gazebo::sensors, 122
CATEGORY_COUNT, 126
CameraSensor_V, 125
CameraSensorPtr, 125
ContactSensor_V, 125
ContactSensorPtr, 125
DepthCameraSensor_V, 125
DepthCameraSensorPtr, 125
ForceTorqueSensorPtr, 125
GpsSensorPtr, 125
GpuRaySensor_V, 125

GpuRaySensorPtr, 125
IMAGE, 126
ImuSensor_V, 125
ImuSensorPtr, 125
MultiCameraSensor_V, 125
MultiCameraSensorPtr, 126
NoisePtr, 126
OTHER, 126
RAY, 126
RFIDSensor_V, 126
RFIDSensorPtr, 126
RFIDTag_V, 126
RFIDTagPtr, 126
RaySensor_V, 126
RaySensorPtr, 126
Sensor_V, 126
SensorCategory, 126
SensorFactoryFn, 126
SensorPtr, 126
SonarSensorPtr, 126
WirelessReceiver_V, 126
WirelessReceiverPtr, 126
WirelessTransceiver_V, 126
WirelessTransceiverPtr, 126
WirelessTransmitter_V, 126
WirelessTransmitterPtr, 126
gazebo::sensors::CameraSensor, 213
~CameraSensor, 214
CameraSensor, 214
Fini, 215
GetCamera, 215
GetImageData, 215
GetImageHeight, 215
GetImageWidth, 215
GetTopic, 215
Init, 215
IsActive, 216
Load, 216
SaveFrame, 216
UpdateImpl, 216
gazebo::sensors::ContactSensor, 267
~ContactSensor, 269
ContactSensor, 269
Fini, 269
GetCollisionContactCount, 269
GetCollisionCount, 269
GetCollisionName, 270
GetContacts, 270
Init, 271
IsActive, 271
Load, 271
UpdateImpl, 271
gazebo::sensors::DepthCameraSensor, 362
~DepthCameraSensor, 363
DepthCameraSensor, 363
Fini, 363
GetDepthCamera, 363
Init, 363
Load, 364
SaveFrame, 364
SetActive, 364
UpdateImpl, 364
gazebo::sensors::ForceTorqueSensor, 418
~ForceTorqueSensor, 420
ConnectUpdate, 420
DisconnectUpdate, 420
Fini, 421
ForceTorqueSensor, 420
GetForce, 421
GetJoint, 421
GetTopic, 421
GetTorque, 421
Init, 421
IsActive, 421
Load, 422
update, 422
UpdateImpl, 422
gazebo::sensors::GpsSensor, 426
~GpsSensor, 427
Fini, 427
GetAltitude, 427
GetLatitude, 428
GetLongitude, 428
GpsSensor, 427
Init, 428
Load, 428
UpdateImpl, 428
gazebo::sensors::GpuRaySensor, 438
~GpuRaySensor, 441
cameraElem, 449
ConnectNewLaserFrame, 442
DisconnectNewLaserFrame, 442
Fini, 442
GetAngleMax, 442
GetAngleMin, 442
GetAngleResolution, 442
GetCameraCount, 442
GetCosHorzFOV, 443
GetCosVertFOV, 443
GetFiducial, 443
GetHorzFOV, 443
GetHorzHalfAngle, 443
GetLaserCamera, 444
GetRange, 444
GetRangeCount, 444
GetRangeCountRatio, 444
GetRangeMax, 444
GetRangeMin, 445

GetRangeResolution, 445
GetRanges, 445
GetRayCount, 445
GetRayCountRatio, 445
GetRetro, 445
GetTopic, 446
GetVertFOV, 446
GetVertHalfAngle, 446
GetVerticalAngleMax, 446
GetVerticalAngleMin, 446
GetVerticalRangeCount, 447
GetVerticalRayCount, 447
GpuRaySensor, 441
horzElem, 449
horzRangeCount, 449
horzRayCount, 449
Init, 447
IsActive, 447
IsHorizontal, 447
Load, 447, 448
rangeCountRatio, 449
rangeElem, 449
scanElem, 449
SetAngleMax, 448
SetAngleMin, 448
SetVerticalAngleMax, 448
SetVerticalAngleMin, 448
UpdateImpl, 449
vertElem, 449
vertRangeCount, 449
vertRayCount, 450
gazebo::sensors::ImuSensor, 480
~ImuSensor, 481
Fini, 481
GetAngularVelocity, 481
GetImuMessage, 482
GetLinearAcceleration, 482
GetOrientation, 482
ImuSensor, 481
Init, 482
IsActive, 482
Load, 482, 483
SetReferencePose, 483
UpdateImpl, 483
gazebo::sensors::MultiCameraSensor, 660
~MultiCameraSensor, 661
Fini, 661
GetCamera, 661
GetCameraCount, 662
GetImageData, 662
GetImageHeight, 662
GetImageWidth, 663
GetTopic, 663
Init, 663
IsActive, 663
Load, 663
MultiCameraSensor, 661
SaveFrame, 664
UpdateImpl, 664
gazebo::sensors::Noise, 689
~Noise, 690
Apply, 690
GAUSSIAN, 690
GAUSSIAN_QUANTIZED, 690
GetBias, 691
GetMean, 691
GetNoiseType, 691
GetStdDev, 691
Load, 691
NONE, 690
Noise, 690
NoiseType, 690
gazebo::sensors::RFIDSensor, 796
~RFIDSensor, 797
AddTag, 797
Fini, 797
Init, 797
Load, 797
RFIDSensor, 797
UpdateImpl, 798
gazebo::sensors::RFIDTag, 798
~RFIDTag, 800
Fini, 800
GetTagPose, 800
Init, 800
Load, 800
RFIDTag, 800
UpdateImpl, 800
gazebo::sensors::RaySensor, 779
~RaySensor, 781
Fini, 781
GetAngleMax, 781
GetAngleMin, 781
GetAngleResolution, 781
GetFiducial, 781
GetLaserShape, 782
GetRange, 782
GetRangeCount, 782
GetRangeMax, 783
GetRangeMin, 783
GetRangeResolution, 783
GetRanges, 783
GetRayCount, 783
GetRetro, 783
GetTopic, 784
GetVerticalAngleMax, 784
GetVerticalAngleMin, 784
GetVerticalRangeCount, 784

GetVerticalRayCount, 784
Init, 785
IsActive, 785
Load, 785
RaySensor, 781
UpdateImpl, 785
gazebo::sensors::Sensor, 837
~Sensor, 840
active, 847
ConnectUpdated, 841
connections, 847
DisconnectUpdated, 841
FillMsg, 841
Fini, 841
GetCategory, 842
GetId, 842
GetLastMeasurementTime, 842
GetLastUpdateTime, 842
GetName, 842
GetParentId, 842
GetParentName, 843
GetPose, 843
GetScopedName, 843
GetTopic, 843
GetType, 843
GetUpdateRate, 844
GetVisualize, 844
GetWorldName, 844
Init, 844
IsActive, 844
lastMeasurementTime, 847
lastUpdateTime, 847
Load, 845
mutexLastUpdateTime, 847
node, 847
parentId, 847
parentName, 847
plugins, 847
pose, 847
poseSub, 847
ResetLastUpdateTime, 845
scene, 848
sdf, 848
Sensor, 840
SetActive, 845
SetParent, 845, 846
SetUpdateRate, 846
Update, 846
UpdateImpl, 846
updatePeriod, 848
world, 848
gazebo::sensors::SensorFactory, 848
GetSensorTypes, 849
NewSensor, 849
RegisterAll, 849
RegisterSensor, 849
gazebo::sensors::SensorManager, 850
CreateSensor, 851
Fini, 851
GetSensor, 852
GetSensorTypes, 852
GetSensors, 852
Init, 852
RemoveSensor, 852
RemoveSensors, 852
ResetLastUpdateTimes, 852
RunThreads, 853
SensorsInitialized, 853
Stop, 853
Update, 853
gazebo::sensors::SonarSensor, 977
~SonarSensor, 979
ConnectUpdate, 979
DisconnectUpdate, 979
Fini, 980
GetRadius, 980
GetRange, 980
GetRangeMax, 980
GetRangeMin, 980
GetTopic, 980
Init, 981
IsActive, 981
Load, 981
SonarSensor, 979
update, 982
UpdateImpl, 981
gazebo::sensors::WirelessReceiver, 1147
~WirelessReceiver, 1149
Fini, 1149
GetMaxFreqFiltered, 1149
GetMinFreqFiltered, 1149
GetSensitivity, 1149
Init, 1149
Load, 1150
WirelessReceiver, 1149
gazebo::sensors::WirelessTransceiver, 1150
~WirelessTransceiver, 1151
Fini, 1152
gain, 1153
GetGain, 1152
GetPower, 1152
GetTopic, 1152
Init, 1152
Load, 1152
parentEntity, 1153
power, 1153
pub, 1153
referencePose, 1153

WirelessTransceiver, 1151
gazebo::sensors::WirelessTransmitter, 1153
~WirelessTransmitter, 1155
freq, 1156
GetESSID, 1155
GetFreq, 1155
GetSignalStrength, 1155
Init, 1156
Load, 1156
ModelStdDesv, 1156
NEmpty, 1156
NObstacle, 1157
UpdateImpl, 1156
WirelessTransmitter, 1155
gazebo::transport, 127
ConnectionPtr, 129
MessagePtr, 129
NodePtr, 129
PublicationPtr, 129
PublicationTransportPtr, 129
PublisherPtr, 129
SubscriberPtr, 129
SubscriptionTransportPtr, 129
gazebo::transport::CallbackHelper, 180
~CallbackHelper, 182
CallbackHelper, 182
GetId, 182
GetLatching, 182
GetMsgType, 182
HandleData, 182
HandleMessage, 183
IsLocal, 183
latching, 183
gazebo::transport::CallbackHelperT
CallbackHelperT, 184
GetMsgType, 185
HandleData, 185
HandleMessage, 185
IsLocal, 186
gazebo::transport::CallbackHelperT< M >, 184
gazebo::transport::Connection, 247
~Connection, 250
AcceptCallback, 250
AsyncRead, 250
Cancel, 250
Connect, 250
ConnectToShutdown, 250
Connection, 250
DisconnectShutdown, 251
EnqueueMsg, 251
GetIPWhiteList, 251
GetId, 251
GetLocalAddress, 252
GetLocalHostname, 252
GetLocalPort, 252
GetLocalURI, 252
GetRemoteAddress, 252
GetRemoteHostname, 252
GetRemotePort, 253
GetRemoteURI, 253
IsOpen, 253
Listen, 253
ProcessWriteQueue, 253
Read, 253
ReadCallback, 250
Shutdown, 254
StartRead, 254
StopRead, 254
ValidateIP, 254
gazebo::transport::ConnectionManager, 254
Advertise, 256
ConnectToRemoteHost, 256
eventConnections, 259
Fini, 256
GetAllPublishers, 256
GetTopicNamespaces, 257
Init, 257
IsRunning, 257
RegisterTopicNamespace, 257
RemoveConnection, 257
Run, 257
Stop, 258
Subscribe, 258
TriggerUpdate, 258
Unadvertise, 258
Unsubscribe, 258
gazebo::transport::IOManager, 494
~IOManager, 495
DecCount, 495
GetCount, 495
GetIO, 495
IOManager, 495
IncCount, 495
Stop, 495
gazebo::transport::Node, 672
~Node, 674
Advertise, 674
DecodeTopicName, 675
EncodeTopicName, 675
Fini, 675
GetId, 675
GetMsgType, 675
GetTopicNamespace, 676
HandleData, 676
HandleMessage, 676
HasLatchedSubscriber, 676
Init, 677
InsertLatchedMsg, 677

Node, 674
ProcessIncoming, 677
ProcessPublishers, 677
Publish, 677
RemoveCallback, 678
Subscribe, 678, 679
gazebo::transport::Publication, 750
~Publication, 751
AddPublisher, 752
AddSubscription, 752
AddTransport, 752
GetCallbackCount, 752
GetLocallyAdvertised, 752
GetMsgType, 753
GetNodeCount, 753
GetRemoteSubscriptionCount, 753
GetTransportCount, 753
HasTransport, 753
LocalPublish, 754
Publication, 751
Publish, 754
RemoveSubscription, 754
RemoveTransport, 754
SetLocallyAdvertised, 754
gazebo::transport::PublicationTransport, 755
~PublicationTransport, 756
AddCallback, 756
Fini, 756
GetConnection, 756
GetMsgType, 756
GetTopic, 756
Init, 756
PublicationTransport, 755
gazebo::transport::Publisher, 757
~Publisher, 758
GetMsgType, 758
GetOutgoingCount, 758
GetPrevMsg, 758
GetPrevMsgPtr, 759
GetTopic, 759
HasConnections, 759
Publish, 759
Publisher, 758
SendMessage, 759
SetNode, 760
SetPublication, 760
WaitForConnection, 760
gazebo::transport::RawCallbackHelper, 776
GetMsgType, 778
HandleData, 778
HandleMessage, 778
IsLocal, 778
RawCallbackHelper, 777
gazebo::transport::SubscribeOptions, 1014
GetLatching, 1015
GetMsgType, 1015
GetNode, 1015
GetTopic, 1015
Init, 1016
SubscribeOptions, 1015
gazebo::transport::Subscriber, 1016
~Subscriber, 1017
GetCallbackId, 1017
GetTopic, 1017
SetCallbackId, 1017
Subscriber, 1017
Unsubscribe, 1017
gazebo::transport::SubscriptionTransport, 1018
~SubscriptionTransport, 1019
GetConnection, 1019
HandleData, 1019
HandleMessage, 1019
Init, 1020
IsLocal, 1020
SubscriptionTransport, 1019
gazebo::transport::TopicManager, 1054
AddNode, 1056
AddNodeToProcess, 1057
Advertise, 1057
ClearBuffers, 1057
ConnectPubToSub, 1057
ConnectSubToPub, 1057
ConnectSubscribers, 1057
DisconnectPubFromSub, 1058
DisconnectSubFromPub, 1058
FindPublication, 1058
Fini, 1058
GetTopicNamespaces, 1058
Init, 1059
IsAdvertised, 1059
PauseIncoming, 1059
ProcessNodes, 1059
Publish, 1059
RegisterTopicNamespace, 1060
RemoveNode, 1060
SubNodeMap, 1056
Subscribe, 1060
Unadvertise, 1060
Unsubscribe, 1060
UpdatePublications, 1061
gazebo::util, 129
DiagnosticTimerPtr, 129
OpenALSinkPtr, 129
OpenALSourcePtr, 130
gazebo::util::DiagnosticManager, 365
GetLabel, 366
GetLogPath, 366
GetTime, 366

GetTimerCount, 367
Init, 367
Lap, 367
StartTimer, 367
StopTimer, 367
gazebo::util::DiagnosticTimer, 368
~DiagnosticTimer, 369
DiagnosticTimer, 369
GetName, 369
Lap, 369
Start, 369
Stop, 369
gazebo::util::LogPlay, 570
GetChunk, 571
GetChunkCount, 571
GetEncoding, 571
GetGazeboVersion, 571
GetHeader, 572
GetLogVersion, 572
GetRandSeed, 572
IsOpen, 572
Open, 572
Step, 573
gazebo::util::LogRecord, 573
Add, 575
Fini, 576
GetBasePath, 576
GetBufferSize, 576
GetEncoding, 576
GetFileSize, 576
GetFilename, 576
GetFirstUpdate, 577
GetPaused, 577
GetRunTime, 577
GetRunning, 577
Init, 577
IsReadyToStart, 578
Notify, 578
Remove, 578
SetBasePath, 578
SetPaused, 578
Start, 579
Stop, 579
Write, 579
gazebo::util::OpenAL, 695
CreateSink, 696
CreateSource, 696
Fini, 697
Load, 697
gazebo::util::OpenALSink, 697
~OpenALSink, 698
OpenALSink, 698
SetPose, 698
SetVelocity, 698
gazebo::util::OpenALSource, 698
~OpenALSource, 699
FillBufferFromFile, 700
FillBufferFromPCM, 700
GetCollisionNames, 700
GetOnContact, 700
HasCollisionName, 700
IsPlaying, 701
Load, 701
OpenALSource, 699
Pause, 701
Play, 701
Rewind, 701
SetGain, 701
SetLoop, 702
SetPitch, 702
SetPose, 702
SetVelocity, 702
Stop, 703
gazebo_core.hh, 1244
GazeboGenerator
google::protobuf::compiler::cpp::GazeboGenerator, 426
GazeboGenerator.hh, 1245
gazeboPathsFromEnv
gazebo::common::SystemPaths, 1029
GenSphericalTexCoord
gazebo::common::Mesh, 609
gazebo::common::MeshManager, 619
gazebo::common::SubMesh, 1009
Generate
google::protobuf::compiler::cpp::GazeboGenerator, 426
generate
gazebo::rendering::GzTerrainMatGen::SM2Profile, 977
generateForCompositeMap
gazebo::rendering::GzTerrainMatGen::SM2Profile, 977
generateFpDynamicShadows
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 860
generateFpDynamicShadowsHelpers
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 860
generateFpDynamicShadowsParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 860
generateFpFooter
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 860
generateFpHeader
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 860

generateFpLayer
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 860
generateFragmentProgram
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 860
generateFragmentProgramSource
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
GenerateShaders
gazebo::rendering::RTShaderSystem, 813
generateVertexProgram
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
generateVertexProgramSource
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
generateVpDynamicShadows
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
generateVpDynamicShadowsParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
generateVpFooter
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
generateVpHeader
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 858
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
GeneratorType
gazebo::math, 108
GeometryFromSDF
Messages, 60
Get
gazebo::common::NodeTransform, 686
get_master_uri
Transport, 94
get_scene
Rendering, 84
get_sensor
Sensors, 90
get_sha1
Common, 40
get_topic_namespaces
Transport, 94
get_world
Classes for physics and dynamics, 69
GetAABB
gazebo::common::Mesh, 609
gazebo::rendering::MovableText, 655
GetAbs
gazebo::math::Vector3, 1096
GetAcceleration
gazebo::physics::LinkState, 566
getAdvertisedTopics
Transport, 95
GetAllPublishers
gazebo::transport::ConnectionManager, 256
GetAltitude
gazebo::sensors::GpsSensor, 427
GetAmbient
gazebo::common::Material, 588
GetAmbientColor
gazebo::rendering::Scene, 821
GetAnchor
gazebo::physics::DARTBallJoint, 281
gazebo::physics::DARTHinge2Joint, 297
gazebo::physics::DARTHingeJoint, 302
gazebo::physics::DARTSliderJoint, 347
gazebo::physics::DARTUniversalJoint, 355
gazebo::physics::Joint, 502
gazebo::physics::ScrewJoint, 833
gazebo::physics::SimbodyBallJoint, 868
gazebo::physics::SimbodyHinge2Joint, 882
gazebo::physics::SimbodyJoint, 894
gazebo::physics::SimbodyUniversalJoint, 947
gazebo::physics::SliderJoint, 974
GetAngle
gazebo::physics::Joint, 503
gazebo::physics::JointState, 525
GetAngleCount
gazebo::physics::BallJoint, 158
gazebo::physics::DARTJoint, 308
gazebo::physics::Hinge2Joint, 472
gazebo::physics::HingeJoint, 474
gazebo::physics::Joint, 503
gazebo::physics::JointState, 525
gazebo::physics::ScrewJoint, 834
gazebo::physics::SliderJoint, 975
gazebo::physics::UniversalJoint, 1065
GetAngleImpl
gazebo::physics::DARTBallJoint, 282
gazebo::physics::DARTHinge2Joint, 297
gazebo::physics::DARTHingeJoint, 302

gazebo::physics::DARTScrewJoint, 342
gazebo::physics::DARTSliderJoint, 347
gazebo::physics::DARTUniversalJoint, 355
gazebo::physics::Joint, 503
gazebo::physics::SimbodyBallJoint, 868
gazebo::physics::SimbodyHinge2Joint, 882
gazebo::physics::SimbodyHingeJoint, 888
gazebo::physics::SimbodyScrewJoint, 932
gazebo::physics::SimbodySliderJoint, 939
gazebo::physics::SimbodyUniversalJoint, 947
GetAngleMax
gazebo::sensors::GpuRaySensor, 442
gazebo::sensors::RaySensor, 781
GetAngleMin
gazebo::sensors::GpuRaySensor, 442
gazebo::sensors::RaySensor, 781
GetAngleResolution
gazebo::sensors::GpuRaySensor, 442
gazebo::sensors::RaySensor, 781
GetAngles
gazebo::physics::JointState, 525
GetAngularDamping
gazebo::physics::Link, 550
GetAngularVelocity
gazebo::sensors::ImuSensor, 481
GetAnimation
gazebo::common::Skeleton, 956
GetAsABGR
gazebo::common::Color, 237
GetAsARGB
gazebo::common::Color, 237
GetAsAxis
gazebo::math::Quaternion, 765
GetAsBGRA
gazebo::common::Color, 237
GetAsEuler
gazebo::math::Quaternion, 766
GetAsHSV
gazebo::common::Color, 237
GetAsMatrix3
gazebo::math::Quaternion, 766
GetAsMatrix4
gazebo::math::Quaternion, 766
GetAsPose
gazebo::math::Matrix4, 601
GetAsRGBA
gazebo::common::Color, 237
GetAsYUV
gazebo::common::Color, 237
GetAspectRatio
gazebo::rendering::Camera, 195
GetAttachedObjectCount
gazebo::rendering::Visual, 1130
GetAttribute
gazebo::physics::DARTJoint, 308
gazebo::physics::Joint, 504
gazebo::physics::SimbodyJoint, 895
GetAutoDisable
gazebo::physics::Model, 629
GetAutoDisableFlag
gazebo::physics::PhysicsEngine, 712
GetAvgColor
gazebo::common::Image, 477
GetAvgFPS
gazebo::rendering::Camera, 195
gazebo::rendering::UserCamera, 1069
gazebo::rendering::WindowManager, 1145
GetAvgHeight
gazebo::rendering::Heightmap, 462
GetAxis
gazebo::physics::SimbodyBallJoint, 868
gazebo::physics::SimbodyHinge2Joint, 882
gazebo::physics::SimbodyUniversalJoint, 947
GetBPP
gazebo::common::Image, 477
GetBackgroundColor
gazebo::rendering::Scene, 821
GetBasePath
gazebo::util::LogRecord, 576
GetBaseline
gazebo::rendering::MovableText, 655
GetBias
gazebo::sensors::Noise, 691
GetBindShapeTransform
gazebo::common::Skeleton, 956
GetBlendFactors
gazebo::common::Material, 588
GetBlendMode
gazebo::common::Material, 588
GetBoundingBox
gazebo::physics::Collision, 223
gazebo::physics::DARTCollision, 288
gazebo::physics::Entity, 382
gazebo::physics::Link, 550
gazebo::physics::Model, 629
gazebo::physics::SimbodyCollision, 874
gazebo::rendering::Visual, 1130
getBoundingRadius
gazebo::rendering::DynamicRenderable, 377
gazebo::rendering::MovableText, 655
GetBufferSize
gazebo::util::LogRecord, 576
GetByName
gazebo::physics::Base, 165
gazebo::physics::World, 1161
GetCallbackCount
gazebo::transport::Publication, 752
GetCallbackId

gazebo::transport::Subscriber, 1017
GetCamera
gazebo::rendering::Scene, 821
gazebo::sensors::CameraSensor, 215
gazebo::sensors::MultiCameraSensor, 661
GetCameraCount
gazebo::rendering::GpuLaser, 433
gazebo::rendering::Scene, 822
gazebo::sensors::GpuRaySensor, 442
gazebo::sensors::MultiCameraSensor, 662
GetCameraToViewportRay
gazebo::rendering::Camera, 196
GetCaptureData
gazebo::rendering::Camera, 196
GetCategory
gazebo::sensors::Sensor, 842
GetCategoryBits
gazebo::physics::DARTCollision, 289
GetCellCount
gazebo::rendering::Grid, 451
GetCellLength
gazebo::rendering::Grid, 451
GetCenter
gazebo::math::Box, 173
GetCharHeight
gazebo::rendering::MovableText, 655
GetChild
gazebo::common::SkeletonNode, 967
gazebo::physics::Base, 165
gazebo::physics::Joint, 504
gazebo::rendering::Visual, 1130
GetChildById
gazebo::common::SkeletonNode, 967
GetChildByName
gazebo::common::SkeletonNode, 968
GetChildCollision
gazebo::physics::Entity, 382
GetChildCount
gazebo::common::SkeletonNode, 968
gazebo::physics::Base, 165
gazebo::rendering::Visual, 1130
GetChildJoints
gazebo::physics::Link, 550
GetChildJointsLinks
gazebo::physics::Link, 550
GetChildLink
gazebo::physics::Entity, 382
GetChunk
gazebo::util::LogPlay, 571
GetChunkCount
gazebo::util::LogPlay, 571
GetCmd
gazebo::common::PID, 723
GetCoG
gazebo::physics::Inertial, 486
GetCollideBits
gazebo::physics::DARTCollision, 289
GetCollision
gazebo::physics::Link, 550, 551
GetCollisionBoundingBox
gazebo::physics::Entity, 383
GetCollisionContactCount
gazebo::sensors::ContactSensor, 269
GetCollisionCount
gazebo::sensors::ContactSensor, 269
GetCollisionName
gazebo::sensors::ContactSensor, 270
GetCollisionNames
gazebo::util::OpenALSource, 700
GetCollisionShape
gazebo::physics::SimbodyCollision, 874
GetCollisionState
gazebo::physics::LinkState, 566
GetCollisionStateCount
gazebo::physics::LinkState, 567
GetCollisionStates
gazebo::physics::LinkState, 567
GetCollisions
gazebo::physics::Link, 551
GetColor
gazebo::rendering::Grid, 452
gazebo::rendering::MovableText, 655
GetConnection
gazebo::transport::PublicationTransport, 756
gazebo::transport::SubscriptionTransport, 1019
GetContact
gazebo::physics::ContactManager, 265
GetContactCount
gazebo::physics::ContactManager, 265
GetContactManager
gazebo::physics::PhysicsEngine, 712
GetContactMaxCorrectingVel
gazebo::physics::PhysicsEngine, 712
GetContactSurfaceLayer
gazebo::physics::PhysicsEngine, 712
GetContacts
gazebo::physics::ContactManager, 265
gazebo::sensors::ContactSensor, 270
GetContactsEnabled
gazebo::physics::Collision, 223
GetCosHorzFOV
gazebo::rendering::GpuLaser, 433
gazebo::sensors::GpuRaySensor, 443
GetCosVertFOV
gazebo::rendering::GpuLaser, 433
gazebo::sensors::GpuRaySensor, 443
GetCount
gazebo::transport::IOManager, 495

GetCurrentDir
SystemPaths.hh, 1387
GetDARTBodyNode
gazebo::physics::DARTCollision, 289
gazebo::physics::DARTLink, 318
GetDARTCollisionShape
gazebo::physics::DARTCollision, 289
GetDARTJoint
gazebo::physics::DARTJoint, 308
GetDARTModel
gazebo::physics::DARTJoint, 308
gazebo::physics::DARTLink, 318
GetDARTPhysics
gazebo::physics::DARTLink, 318
gazebo::physics::DARTModel, 328
GetDARTSkeleton
gazebo::physics::DARTModel, 328
GetDARTWorld
gazebo::physics::DARTLink, 318
gazebo::physics::DARTModel, 328
gazebo::physics::DARTPhysics, 333
GetDBConfig
Common, 41
GetDamping
gazebo::physics::Joint, 504
GetDampingCoefficient
gazebo::physics::Joint, 504
GetData
gazebo::common::Image, 477
GetDblNormal
gazebo::math::Rand, 775
GetDblUniform
gazebo::math::Rand, 775
GetDepthCamera
gazebo::sensors::DepthCameraSensor, 363
GetDepthData
gazebo::rendering::DepthCamera, 361
GetDepthWrite
gazebo::common::Material, 589
GetDiffuse
gazebo::common::Material, 589
GetDiffuseColor
gazebo::rendering::Light, 537
GetDirection
gazebo::rendering::Camera, 196
gazebo::rendering::Light, 537
GetDirtyPose
gazebo::physics::Entity, 383
GetDistToLine
gazebo::math::Vector3, 1096
GetDynamicsWorld
gazebo::physics::SimbodyPhysics, 919
GetESSID
gazebo::sensors::WirelessTransmitter, 1155
GetEffectiveMassProps
gazebo::physics::SimbodyLink, 905
GetEffortLimit
gazebo::physics::Joint, 504
GetElapsed
gazebo::common::Timer, 1054
GetElevationReference
gazebo::common::SphericalCoordinates, 991
GetEmissive
gazebo::common::Material, 589
GetEnablePhysicsEngine
gazebo::physics::World, 1161
GetEnabled
gazebo::physics::DARTLink, 318
gazebo::physics::Link, 551
gazebo::physics::SimbodyLink, 905
GetEncoding
gazebo::util::LogPlay, 571
gazebo::util::LogRecord, 576
GetEntity
gazebo::physics::World, 1161
GetEntityBelowPoint
gazebo::physics::World, 1161
GetErrorFile
gazebo::common::Exception, 417
GetErrorStr
gazebo::common::Exception, 418
GetErrors
gazebo::common::PID, 723
GetEulerRotation
gazebo::math::Matrix4, 602
GetExp
gazebo::math::Quaternion, 766
GetFarClip
gazebo::rendering::Camera, 196
gazebo::rendering::GpuLaser, 433
GetFiducial
gazebo::physics::MultiRayShape, 668
gazebo::physics::RayShape, 788
gazebo::sensors::GpuRaySensor, 443
gazebo::sensors::RaySensor, 781
GetFile
gazebo::common::AudioDecoder, 154
GetFileSize
gazebo::util::LogRecord, 576
GetFilename
gazebo::common::Image, 477
gazebo::PluginT, 734
gazebo::util::LogRecord, 576
GetFirstContact
gazebo::rendering::Scene, 822
GetFirstUpdate
gazebo::util::LogRecord, 577
GetFocalPoint

gazebo::rendering::OrbitViewController, 704
GetFont
gazebo::rendering::MovableText, 656
GetForce
gazebo::physics::DARTJoint, 308
gazebo::physics::Joint, 505
gazebo::physics::SimbodyJoint, 895
gazebo::sensors::ForceTorqueSensor, 421
GetForceTorque
gazebo::physics::DARTJoint, 309
gazebo::physics::Joint, 505
gazebo::physics::SimbodyJoint, 895
GetForceTorqueWithAppliedForce
Joint_TEST, 518
GetFrameAt
gazebo::common::NodeAnimation, 681
GetFrameCount
gazebo::common::NodeAnimation, 681
GetFrameFilename
gazebo::rendering::Camera, 196
GetFreq
gazebo::sensors::WirelessTransmitter, 1155
GetGUIOverlay
gazebo::rendering::UserCamera, 1069
GetGain
gazebo::sensors::WirelessTransceiver, 1152
GetGazeboPaths
gazebo::common::SystemPaths, 1028
GetGazeboVersion
gazebo::util::LogPlay, 571
GetGlobalAxis
gazebo::physics::DARTBallJoint, 282
gazebo::physics::DARTHinge2Joint, 297
gazebo::physics::DARTHingeJoint, 302
gazebo::physics::DARTScrewJoint, 342
gazebo::physics::DARTSliderJoint, 347
gazebo::physics::DARTUniversalJoint, 355
gazebo::physics::Joint, 505
gazebo::physics::SimbodyBallJoint, 868
gazebo::physics::SimbodyHinge2Joint, 882
gazebo::physics::SimbodyHingeJoint, 888
gazebo::physics::SimbodyScrewJoint, 932
gazebo::physics::SimbodySliderJoint, 939
gazebo::physics::SimbodyUniversalJoint, 947
GetGlobalPoints
gazebo::physics::RayShape, 788
GetGranularity
gazebo::physics::MapShape, 581
GetGravity
gazebo::physics::PhysicsEngine, 712
GetGravityMode
gazebo::physics::DARTLink, 319
gazebo::physics::Link, 551
gazebo::physics::SimbodyLink, 905
GetGrid
gazebo::rendering::Scene, 822
GetGridCount
gazebo::rendering::Scene, 822
GetGripper
gazebo::physics::Model, 629
GetGripperCount
gazebo::physics::Model, 630
GetHFOV
gazebo::rendering::Camera, 196
GetHandle
gazebo::common::SkeletonNode, 968
gazebo::PluginT, 734
GetHeader
gazebo::util::LogPlay, 572
Messages, 60
GetHeadingOffset
gazebo::common::SphericalCoordinates, 991
GetHeight
gazebo::common::Image, 477
gazebo::common::Video, 1115
gazebo::physics::HeightmapShape, 468
gazebo::physics::MapShape, 581
gazebo::rendering::Grid, 452
gazebo::rendering::Heightmap, 462
GetHeightBelowPoint
gazebo::rendering::Scene, 822
GetHeightmap
gazebo::rendering::Scene, 823
GetHighStop
gazebo::physics::BallJoint, 158
gazebo::physics::DARTJoint, 309
gazebo::physics::Joint, 506
gazebo::physics::SimbodyHinge2Joint, 882
gazebo::physics::SimbodyHingeJoint, 888
gazebo::physics::SimbodyScrewJoint, 932
gazebo::physics::SimbodySliderJoint, 939
gazebo::physics::SimbodyUniversalJoint, 947
GetHorzFOV
gazebo::rendering::GpuLaser, 433
gazebo::sensors::GpuRaySensor, 443
GetHorzHalfAngle
gazebo::rendering::GpuLaser, 433
gazebo::sensors::GpuRaySensor, 443
GetIO
gazebo::transport::IOManager, 495
GetIPWhiteList
gazebo::transport::Connection, 251
GetIXX
gazebo::physics::Inertial, 486
GetIXY
gazebo::physics::Inertial, 487
GetIXZ
gazebo::physics::Inertial, 487

GetIYY
gazebo::physics::Inertial, 487
GetIYZ
gazebo::physics::Inertial, 487
GetIZZ
gazebo::physics::Inertial, 487
GetId
gazebo::common::SkeletonNode, 968
gazebo::event::Connection, 247
gazebo::physics::Base, 166
gazebo::rendering::Scene, 823
gazebo::rendering::Visual, 1130
gazebo::sensors::Sensor, 842
gazebo::transport::CallbackHelper, 182
gazebo::transport::Connection, 251
gazebo::transport::Node, 675
GetIdString
gazebo::rendering::Scene, 823
GetImage
gazebo::physics::HeightmapShape, 468
gazebo::rendering::Heightmap, 463
GetImageByteSize
gazebo::rendering::Camera, 197
GetImageData
gazebo::rendering::Camera, 197
gazebo::sensors::CameraSensor, 215
gazebo::sensors::MultiCameraSensor, 662
GetImageDepth
gazebo::rendering::Camera, 197
GetImageFormat
gazebo::rendering::Camera, 198
GetImageHeight
gazebo::rendering::Camera, 198
gazebo::rendering::UserCamera, 1070
gazebo::sensors::CameraSensor, 215
gazebo::sensors::MultiCameraSensor, 662
GetImageWidth
gazebo::rendering::Camera, 198
gazebo::rendering::UserCamera, 1070
gazebo::sensors::CameraSensor, 215
gazebo::sensors::MultiCameraSensor, 663
GetImuMessage
gazebo::sensors::ImuSensor, 482
GetIndex
gazebo::common::SubMesh, 1009
GetIndexCount
gazebo::common::Mesh, 609
gazebo::common::SubMesh, 1009
GetInertiaRatio
gazebo::physics::Joint, 506
GetInertial
gazebo::physics::Inertial, 486
gazebo::physics::Link, 551
GetInitialAnchorPose
gazebo::physics::Joint, 506
GetInitialRelativePose
gazebo::physics::Entity, 383
GetInitialized
gazebo::rendering::Camera, 198
gazebo::rendering::Scene, 823
gazebo::Server, 856
GetIntNormal
gazebo::math::Rand, 775
GetIntUniform
gazebo::math::Rand, 776
GetInterpolatedKeyFrame
gazebo::common::NumericAnimation, 693
gazebo::common::PoseAnimation, 745
GetIntersection
DART Physics, 75
gazebo::physics::RayShape, 789
gazebo::physics::SimbodyRayShape, 928
GetInverse
gazebo::math::Pose, 739
gazebo::math::Quaternion, 766
GetInverseBindTransform
gazebo::common::SkeletonNode, 968
GetJoint
gazebo::physics::Model, 630
gazebo::sensors::ForceTorqueSensor, 421
GetJointController
gazebo::physics::Model, 630
GetJointCount
gazebo::physics::Model, 630
GetJointLink
gazebo::physics::DARTJoint, 309
gazebo::physics::Joint, 507
gazebo::physics::SimbodyJoint, 896
GetJointState
gazebo::physics::ModelState, 644
GetJointStateCount
gazebo::physics::ModelState, 645
GetJointStates
gazebo::physics::ModelState, 645
GetJoints
gazebo::physics::Model, 630
GetKeyFrame
gazebo::common::Animation, 147
gazebo::common::NodeAnimation, 681, 682
GetKeyFrameCount
gazebo::common::Animation, 147
GetKeyFramesAtTime
gazebo::common::Animation, 148
GetKinematic
gazebo::physics::DARTLink, 319
gazebo::physics::Link, 552
GetLabel
gazebo::util::DiagnosticManager, 366

GetLaserCamera
gazebo::sensors::GpuRaySensor, 444
GetLaserData
gazebo::rendering::GpuLaser, 434
GetLaserRetro
gazebo::physics::Collision, 223
GetLaserShape
gazebo::sensors::RaySensor, 782
GetLastMeasurementTime
gazebo::sensors::Sensor, 842
GetLastRenderWallTime
gazebo::rendering::Camera, 198
GetLastUpdateTime
gazebo::sensors::Sensor, 842
GetLatching
gazebo::transport::CallbackHelper, 182
gazebo::transport::SubscribeOptions, 1015
GetLatitude
gazebo::sensors::GpsSensor, 428
GetLatitudeReference
gazebo::common::SphericalCoordinates, 991
GetLength
gazebo::common::Animation, 148
gazebo::common::NodeAnimation, 682
gazebo::common::SkeletonAnimation, 961
gazebo::math::Vector3, 1097
gazebo::math::Vector4, 1108
gazebo::physics::CylinderShape, 277
gazebo::physics::RayShape, 789
GetLight
gazebo::rendering::Scene, 823, 824
GetLightCount
gazebo::rendering::Scene, 824
GetLighting
gazebo::common::Material, 589
getLights
gazebo::rendering::MovableText, 656
GetLineWidth
gazebo::rendering::Grid, 452
GetLinearAcceleration
gazebo::sensors::ImuSensor, 482
GetLinearDamping
gazebo::physics::Link, 552
GetLink
gazebo::physics::Collision, 224
gazebo::physics::Model, 630
GetLinkForce
gazebo::physics::DARTJoint, 310
gazebo::physics::Joint, 507
gazebo::physics::SimbodyJoint, 896
GetLinkState
gazebo::physics::ModelState, 645
GetLinkStateCount
gazebo::physics::ModelState, 646
GetLinkStates
gazebo::physics::ModelState, 646
GetLinkTorque
gazebo::physics::DARTJoint, 310
gazebo::physics::Joint, 507
gazebo::physics::SimbodyJoint, 896
GetLinks
gazebo::physics::Model, 631
GetLocalAddress
gazebo::transport::Connection, 252
GetLocalAxis
gazebo::physics::Joint, 507
GetLocalHostname
gazebo::transport::Connection, 252
GetLocalPort
gazebo::transport::Connection, 252
GetLocalURI
gazebo::transport::Connection, 252
GetLocallyAdvertised
gazebo::transport::Publication, 752
GetLog
gazebo::math::Quaternion, 766
GetLogPath
gazebo::common::SystemPaths, 1028
gazebo::util::DiagnosticManager, 366
GetLogVersion
gazebo::util::LogPlay, 572
GetLongitude
gazebo::sensors::GpsSensor, 428
GetLongitudeReference
gazebo::common::SphericalCoordinates, 991
GetLowStop
gazebo::physics::BallJoint, 158
gazebo::physics::DARTJoint, 310
gazebo::physics::Joint, 508
gazebo::physics::SimbodyHinge2Joint, 883
gazebo::physics::SimbodyHingeJoint, 888
gazebo::physics::SimbodyScrewJoint, 932
gazebo::physics::SimbodySliderJoint, 939
gazebo::physics::SimbodyUniversalJoint, 948
GetLowerLimit
gazebo::physics::Joint, 508
GetMOI
gazebo::physics::Inertial, 487, 488
GetManager
gazebo::rendering::Scene, 824
GetMass
gazebo::physics::Inertial, 487
GetMassProperties
gazebo::physics::SimbodyLink, 905
GetMaterial
gazebo::common::Mesh, 609
getMaterial
gazebo::rendering::MovableText, 656

GetMaterialCount
gazebo::common::Mesh, 610
GetMaterialIndex
gazebo::common::SubMesh, 1009
GetMaterialName
gazebo::rendering::Visual, 1130
GetMax
gazebo::common::Mesh, 610
gazebo::common::SubMesh, 1009
gazebo::math::Vector3, 1097
GetMaxAngle
gazebo::physics::MultiRayShape, 668
GetMaxColor
gazebo::common::Image, 478
GetMaxContacts
gazebo::physics::Collision, 224
gazebo::physics::PhysicsEngine, 713
GetMaxForce
gazebo::physics::DARTBallJoint, 282
gazebo::physics::DARTHinge2Joint, 297
gazebo::physics::DARTHingeJoint, 302
gazebo::physics::DARTScrewJoint, 342
gazebo::physics::DARTSliderJoint, 348
gazebo::physics::DARTUniversalJoint, 355
gazebo::physics::Joint, 508
gazebo::physics::SimbodyBallJoint, 868
gazebo::physics::SimbodyHinge2Joint, 883
gazebo::physics::SimbodyHingeJoint, 889
gazebo::physics::SimbodyScrewJoint, 933
gazebo::physics::SimbodySliderJoint, 940
gazebo::physics::SimbodyUniversalJoint, 948
GetMaxFreqFiltered
gazebo::sensors::WirelessReceiver, 1149
GetMaxHeight
gazebo::physics::HeightmapShape, 468
GetMaxIndex
gazebo::common::SubMesh, 1009
GetMaxRange
gazebo::physics::MultiRayShape, 669
GetMaxStepSize
gazebo::physics::PhysicsEngine, 713
GetMean
gazebo::sensors::Noise, 691
GetMesh
gazebo::common::MeshManager, 619
GetMeshAABB
gazebo::common::MeshManager, 620
GetMeshName
gazebo::rendering::Visual, 1131
GetMeshURI
gazebo::physics::MeshShape, 622
GetMin
gazebo::common::Mesh, 610
gazebo::common::SubMesh, 1009
gazebo::math::Vector3, 1097
GetMinAngle
gazebo::physics::MultiRayShape, 669
GetMinFreqFiltered
gazebo::sensors::WirelessReceiver, 1149
GetMinHeight
gazebo::physics::HeightmapShape, 468
GetMinRange
gazebo::physics::MultiRayShape, 669
getMinimalComms
Transport, 95
GetMode
Rendering, 84
GetModel
gazebo::physics::Collision, 224
gazebo::physics::Link, 552
gazebo::physics::World, 1162
GetModelBelowPoint
gazebo::physics::World, 1162
GetModelConfig
Common, 41
GetModelCount
gazebo::physics::World, 1163
GetModelFile
Common, 41
GetModelName
Common, 41
GetModelPath
Common, 42
GetModelPaths
gazebo::common::SystemPaths, 1028
GetModelState
gazebo::physics::WorldState, 1173
GetModelStateCount
gazebo::physics::WorldState, 1173
GetModelStates
gazebo::physics::WorldState, 1173, 1174
GetModelTransform
gazebo::common::SkeletonNode, 968
GetModelVisualAt
gazebo::rendering::Scene, 824
GetModels
Common, 42
gazebo::physics::World, 1163
GetMouseHit
gazebo::rendering::Heightmap, 463
GetMovableType
gazebo::rendering::DynamicLines, 373
gazebo::rendering::DynamicRenderable, 377
getMovableType
gazebo::rendering::DynamicLines, 373
GetMsgType
gazebo::transport::CallbackHelper, 182
gazebo::transport::CallbackHelperT, 185

gazebo::transport::Node, 675
gazebo::transport::Publication, 753
gazebo::transport::PublicationTransport, 756
gazebo::transport::Publisher, 758
gazebo::transport::RawCallbackHelper, 778
gazebo::transport::SubscribeOptions, 1015
GetMsgTypes
gazebo::msgs::MsgFactory, 659
GetName
gazebo::common::Material, 589
gazebo::common::Mesh, 610
gazebo::common::NodeAnimation, 682
gazebo::common::SkeletonAnimation, 962
gazebo::common::SkeletonNode, 969
gazebo::common::SubMesh, 1010
gazebo::physics::Base, 166
gazebo::physics::Gripper, 454
gazebo::physics::State, 1000
gazebo::physics::World, 1163
gazebo::rendering::Camera, 198
gazebo::rendering::Light, 538
gazebo::rendering::Scene, 824
gazebo::rendering::Visual, 1131
gazebo::sensors::Sensor, 842
gazebo::util::DiagnosticTimer, 369
GetNearClip
gazebo::rendering::Camera, 199
gazebo::rendering::GpuLaser, 434
GetNearestEntityBelow
gazebo::physics::Entity, 383
GetNextFrame
gazebo::common::Video, 1115
GetNode
gazebo::transport::SubscribeOptions, 1015
GetNodeAssignment
gazebo::common::SubMesh, 1010
GetNodeAssignmentsCount
gazebo::common::SubMesh, 1010
GetNodeByHandle
gazebo::common::Skeleton, 956
GetNodeById
gazebo::common::Skeleton, 956
GetNodeByName
gazebo::common::Skeleton, 957
GetNodeCount
gazebo::common::SkeletonAnimation, 962
gazebo::transport::Publication, 753
GetNodePoseAt
gazebo::common::SkeletonAnimation, 962
GetNodes
gazebo::common::Skeleton, 957
GetNoiseType
gazebo::sensors::Noise, 691
GetNormal
gazebo::common::SubMesh, 1010
gazebo::math::Vector3, 1097
gazebo::physics::PlaneShape, 731
GetNormalCount
gazebo::common::Mesh, 610
gazebo::common::SubMesh, 1010
GetNormalMap
gazebo::rendering::Visual, 1131
GetNumAnimations
gazebo::common::Skeleton, 957
GetNumJoints
gazebo::common::Skeleton, 957
GetNumNodes
gazebo::common::Skeleton, 957
GetNumPoints
gazebo::math::RotationSpline, 808
GetNumRawTrans
gazebo::common::SkeletonNode, 969
GetNumVertNodeWeights
gazebo::common::Skeleton, 957
GetOgreCamera
gazebo::rendering::Camera, 199
GetOgrePaths
gazebo::common::SystemPaths, 1028
GetOgreTerrain
gazebo::rendering::Heightmap, 463
GetOnContact
gazebo::util::OpenALSource, 700
GetOperationType
gazebo::rendering::DynamicRenderable, 377
GetOrientation
gazebo::sensors::ImuSensor, 482
GetOutgoingCount
gazebo::transport::Publisher, 758
GetPSSMShadowCameraSetup
gazebo::rendering::RTShaderSystem, 813
GetParam
gazebo::physics::DARTPhysics, 333
gazebo::physics::PhysicsEngine, 713
GetParent
gazebo::common::SkeletonNode, 969
gazebo::physics::Base, 166
gazebo::physics::Joint, 509
gazebo::rendering::Projector, 749
gazebo::rendering::Visual, 1131
GetParentId
gazebo::physics::Base, 166
gazebo::sensors::Sensor, 842
GetParentJoints
gazebo::physics::Link, 552
GetParentJointsLinks
gazebo::physics::Link, 552
GetParentModel
gazebo::physics::Entity, 383

GetParentName
gazebo::sensors::Sensor, 843
GetPath
gazebo::common::Mesh, 610
GetPauseTime
gazebo::physics::World, 1163
GetPaused
gazebo::util::LogRecord, 577
GetPerpendicular
gazebo::math::Vector3, 1097
GetPhysicsEngine
gazebo::physics::World, 1163
GetPhysicsUpdateMutex
gazebo::physics::PhysicsEngine, 713
GetPitch
gazebo::common::Image, 478
gazebo::math::Quaternion, 767
GetPitchNode
gazebo::rendering::Camera, 199
GetPixel
gazebo::common::Image, 478
GetPixelFormat
gazebo::common::Image, 478
GetPluginCount
gazebo::physics::Model, 631
GetPluginPaths
gazebo::common::SystemPaths, 1029
GetPoint
gazebo::math::RotationSpline, 808
gazebo::math::Spline, 995
gazebo::rendering::DynamicLines, 373
GetPointCount
gazebo::math::Spline, 995
gazebo::rendering::DynamicLines, 374
GetPointSize
gazebo::common::Material, 589
GetPos
gazebo::physics::HeightmapShape, 469
GetPose
gazebo::physics::CollisionState, 231
gazebo::physics::Inertial, 488
gazebo::physics::LinkState, 567
gazebo::physics::ModelState, 646
gazebo::physics::SimbodyPhysics, 919
gazebo::rendering::Visual, 1131
gazebo::sensors::Sensor, 843
GetPoseAt
gazebo::common::SkeletonAnimation, 962
GetPoseAtX
gazebo::common::SkeletonAnimation, 963
GetPosition
gazebo::rendering::Light, 538
gazebo::rendering::Visual, 1131
GetPower
gazebo::sensors::WirelessTransceiver, 1152
GetPrevMsg
gazebo::transport::Publisher, 758
GetPrevMsgPtr
gazebo::transport::Publisher, 759
GetPrimitiveType
gazebo::common::SubMesh, 1010
GetPrincipalMoments
gazebo::physics::Inertial, 488
GetProductsofInertia
gazebo::physics::Inertial, 488
GetQuiet
Common, 42
GetRGBData
gazebo::common::Image, 478
GetRadius
gazebo::physics::CylinderShape, 277
gazebo::physics::SphereShape, 987
gazebo::sensors::SonarSensor, 980
GetRandSeed
gazebo::util::LogPlay, 572
GetRange
gazebo::physics::MultiRayShape, 669
gazebo::sensors::GpuRaySensor, 444
gazebo::sensors::RaySensor, 782
gazebo::sensors::SonarSensor, 980
GetRangeCount
gazebo::sensors::GpuRaySensor, 444
gazebo::sensors::RaySensor, 782
GetRangeCountRatio
gazebo::sensors::GpuRaySensor, 444
GetRangeMax
gazebo::sensors::GpuRaySensor, 444
gazebo::sensors::RaySensor, 783
gazebo::sensors::SonarSensor, 980
GetRangeMin
gazebo::sensors::GpuRaySensor, 445
gazebo::sensors::RaySensor, 783
gazebo::sensors::SonarSensor, 980
GetRangeResolution
gazebo::sensors::GpuRaySensor, 445
gazebo::sensors::RaySensor, 783
GetRanges
gazebo::sensors::GpuRaySensor, 445
gazebo::sensors::RaySensor, 783
GetRawTransform
gazebo::common::SkeletonNode, 969
GetRawTransforms
gazebo::common::SkeletonNode, 969
GetRayCount
gazebo::sensors::GpuRaySensor, 445
gazebo::sensors::RaySensor, 783
GetRayCountRatio
gazebo::rendering::GpuLaser, 434

gazebo::sensors::GpuRaySensor, 445
GetRealTime
gazebo::physics::State, 1000
gazebo::physics::World, 1163
GetRealTimeUpdateRate
gazebo::physics::PhysicsEngine, 713
GetRelativeAngularAccel
gazebo::physics::Collision, 224
gazebo::physics::Entity, 384
gazebo::physics::Link, 552
gazebo::physics::Model, 631
GetRelativeAngularVel
gazebo::physics::Collision, 224
gazebo::physics::Entity, 384
gazebo::physics::Link, 553
gazebo::physics::Model, 631
GetRelativeForce
gazebo::physics::Link, 553
GetRelativeLinearAccel
gazebo::physics::Collision, 224
gazebo::physics::Entity, 384
gazebo::physics::Link, 553
gazebo::physics::Model, 631
GetRelativeLinearVel
gazebo::physics::Collision, 225
gazebo::physics::Entity, 384
gazebo::physics::Link, 553
gazebo::physics::Model, 632
GetRelativePoints
gazebo::physics::RayShape, 789
GetRelativePose
gazebo::physics::Entity, 384
GetRelativeTorque
gazebo::physics::Link, 553
GetRemoteAddress
gazebo::transport::Connection, 252
GetRemoteHostname
gazebo::transport::Connection, 252
GetRemotePort
gazebo::transport::Connection, 253
GetRemoteSubscriptionCount
gazebo::transport::Publication, 753
GetRemoteURI
gazebo::transport::Connection, 253
getRenderOperation
gazebo::rendering::MovableText, 656
GetRenderPathType
gazebo::rendering::RenderEngine, 794
GetRenderRate
gazebo::rendering::Camera, 199
GetRenderTexture
gazebo::rendering::Camera, 199
GetResRange
gazebo::physics::MultiRayShape, 669
GetRetro
gazebo::physics::MultiRayShape, 669
gazebo::physics::RayShape, 789
gazebo::sensors::GpuRaySensor, 445
gazebo::sensors::RaySensor, 783
GetRight
gazebo::rendering::Camera, 199
GetRoll
gazebo::math::Quaternion, 767
GetRootNode
gazebo::common::Skeleton, 958
GetRootVisual
gazebo::rendering::Visual, 1132
GetRotation
gazebo::common::PoseKeyFrame, 747
gazebo::math::Matrix4, 602
gazebo::rendering::Visual, 1132
GetRounded
gazebo::math::Vector3, 1098
GetRunTime
gazebo::util::LogRecord, 577
GetRunning
gazebo::common::Timer, 1054
gazebo::physics::World, 1164
gazebo::util::LogRecord, 577
GetSDF
gazebo::physics::Actor, 134
gazebo::physics::Base, 167
gazebo::physics::Model, 632
GetSID
gazebo::common::NodeTransform, 686
GetSORPGSIters
gazebo::physics::PhysicsEngine, 714
GetSORPGSPreconIters
gazebo::physics::PhysicsEngine, 714
GetSORPGSW
gazebo::physics::PhysicsEngine, 714
GetSampleCount
gazebo::physics::MultiRayShape, 670
GetSampleRate
gazebo::common::AudioDecoder, 154
GetSaveable
gazebo::physics::Base, 166
GetScale
gazebo::physics::MapShape, 582
gazebo::physics::Shape, 863
gazebo::rendering::Visual, 1132
GetScanResolution
gazebo::physics::MultiRayShape, 670
GetScene
gazebo::rendering::Camera, 200
gazebo::rendering::RenderEngine, 794
gazebo::rendering::Visual, 1132
GetSceneCount

gazebo::rendering::RenderEngine, 794
GetSceneNode
gazebo::rendering::Camera, 200
gazebo::rendering::Grid, 452
gazebo::rendering::Visual, 1132
GetScopedName
gazebo::physics::Base, 166
gazebo::sensors::Sensor, 843
GetScreenshotPath
gazebo::rendering::Camera, 200
GetSeed
gazebo::math::Rand, 776
GetSelectedEntity
gazebo::physics::World, 1164
GetSelectedVisual
gazebo::rendering::Scene, 825
GetSelfCollide
gazebo::physics::Link, 554
GetSensitivity
gazebo::sensors::WirelessReceiver, 1149
GetSensor
gazebo::sensors::SensorManager, 852
GetSensorCount
gazebo::physics::Link, 554
gazebo::physics::Model, 632
GetSensorName
gazebo::physics::Link, 554
GetSensorTypes
gazebo::sensors::SensorFactory, 849
gazebo::sensors::SensorManager, 852
GetSensors
gazebo::sensors::SensorManager, 852
GetSetWorldPoseMutex
gazebo::physics::World, 1164
GetShadeMode
gazebo::common::Material, 590
GetShaderType
gazebo::rendering::Visual, 1132
GetShadowsEnabled
gazebo::rendering::Scene, 825
GetShape
gazebo::physics::Collision, 225
GetShapeType
gazebo::physics::Collision, 225
GetShininess
gazebo::common::Material, 590
GetShowClouds
gazebo::rendering::Scene, 825
GetShowOnTop
gazebo::rendering::MovableText, 656
GetSignalStrength
gazebo::sensors::WirelessTransmitter, 1155
GetSimTime
gazebo::physics::State, 1000
gazebo::physics::World, 1164
gazebo::rendering::Scene, 825
GetSize
gazebo::math::Box, 173
gazebo::physics::BoxShape, 178
gazebo::physics::HeightmapShape, 469
gazebo::physics::MeshShape, 623
gazebo::physics::PlaneShape, 731
GetSkeleton
gazebo::common::Mesh, 610
GetSpaceWidth
gazebo::rendering::MovableText, 656
GetSpecular
gazebo::common::Material, 590
GetSpecularColor
gazebo::rendering::Light, 538
GetSphericalCoordinates
gazebo::physics::World, 1164
GetSquaredLength
gazebo::math::Vector3, 1098
gazebo::math::Vector4, 1108
getSquaredViewDepth
gazebo::rendering::DynamicRenderable, 377
gazebo::rendering::MovableText, 656
GetStartTime
gazebo::physics::World, 1164
GetState
gazebo::physics::Collision, 225
Rendering, 84
GetStdDev
gazebo::sensors::Noise, 691
GetSubMesh
gazebo::common::Mesh, 611
GetSubMeshCount
gazebo::common::Mesh, 611
GetSubMeshName
gazebo::rendering::Visual, 1133
GetSubSampling
gazebo::physics::HeightmapShape, 469
GetSum
gazebo::math::Vector3, 1098
GetSurface
gazebo::physics::Collision, 225
GetSurfaceType
gazebo::common::SphericalCoordinates, 991
GetTagPose
gazebo::sensors::RFIDTag, 800
GetTangent
gazebo::math::Spline, 995
GetTargetRealTimeFactor
gazebo::physics::PhysicsEngine, 714
GetTension
gazebo::math::Spline, 995
GetTerrainSubdivisionCount

gazebo::rendering::Heightmap, 463
GetTexCoord
gazebo::common::SubMesh, 1011
GetTexCoordCount
gazebo::common::Mesh, 611
gazebo::common::SubMesh, 1011
GetText
gazebo::rendering::MovableText, 656
GetTextureHeight
gazebo::rendering::Camera, 200
GetTextureImage
gazebo::common::Material, 590
GetTextureWidth
gazebo::rendering::Camera, 200
GetThreadPitch
gazebo::physics::DARTScrewJoint, 342
gazebo::physics::ScrewJoint, 834
gazebo::physics::SimbodyScrewJoint, 933
GetThreshold
gazebo::physics::MapShape, 582
GetTime
gazebo::common::Animation, 148
gazebo::common::KeyFrame, 533
gazebo::util::DiagnosticManager, 366
GetTimeAtX
gazebo::common::NodeAnimation, 682
GetTimerCount
gazebo::util::DiagnosticManager, 367
GetTopic
gazebo::sensors::CameraSensor, 215
gazebo::sensors::ForceTorqueSensor, 421
gazebo::sensors::GpuRaySensor, 446
gazebo::sensors::MultiCameraSensor, 663
gazebo::sensors::RaySensor, 784
gazebo::sensors::Sensor, 843
gazebo::sensors::SonarSensor, 980
gazebo::sensors::WirelessTransceiver, 1152
gazebo::transport::PublicationTransport, 756
gazebo::transport::Publisher, 759
gazebo::transport::SubscribeOptions, 1015
gazebo::transport::Subscriber, 1017
getTopicMsgType
Transport, 95
GetTopicNamespace
gazebo::transport::Node, 676
GetTopicNamespaces
gazebo::transport::ConnectionManager, 257
gazebo::transport::TopicManager, 1058
GetTorque
gazebo::sensors::ForceTorqueSensor, 421
GetTransform
gazebo::common::SkeletonNode, 970
GetTransforms
gazebo::common::SkeletonNode, 970
GetTranslation
gazebo::common::PoseKeyFrame, 747
gazebo::math::Matrix4, 602
GetTransparency
gazebo::common::Material, 590
gazebo::rendering::Visual, 1133
GetTransportCount
gazebo::transport::Publication, 753
GetTriangleCount
gazebo::rendering::Camera, 200
gazebo::rendering::UserCamera, 1070
gazebo::rendering::WindowManager, 1145
GetType
gazebo::common::NodeTransform, 687
gazebo::physics::Base, 167
gazebo::physics::DARTPhysics, 334
gazebo::physics::PhysicsEngine, 714
gazebo::physics::SimbodyPhysics, 920
gazebo::PluginT, 734
gazebo::rendering::Light, 538
gazebo::sensors::Sensor, 843
GetTypeString
gazebo::physics::SimbodyPhysics, 920
gazebo::rendering::FPSViewController, 424
gazebo::rendering::OrbitViewController, 704
gazebo::rendering::ViewController, 1119
GetURI
Common, 42
gazebo::physics::HeightmapShape, 469
gazebo::physics::MapShape, 582
getUniqueId
Classes for physics and dynamics, 70
GetUp
gazebo::rendering::Camera, 201
GetUpdatePeriod
gazebo::physics::PhysicsEngine, 714
GetUpdateRate
gazebo::sensors::Sensor, 844
GetUpperLimit
gazebo::physics::Joint, 509
GetUserCamera
gazebo::rendering::Scene, 825
GetUserCameraCount
gazebo::rendering::Scene, 826
GetVFOV
gazebo::rendering::Camera, 201
GetValue
gazebo::common::NumericKeyFrame, 695
GetVelocity
gazebo::physics::DARTBallJoint, 283
gazebo::physics::DARTHinge2Joint, 298
gazebo::physics::DARTHingeJoint, 303
gazebo::physics::DARTScrewJoint, 343
gazebo::physics::DARTSliderJoint, 348

gazebo::physics::DARTUniversalJoint, 356
gazebo::physics::Joint, 509
gazebo::physics::LinkState, 567
gazebo::physics::SimbodyBallJoint, 869
gazebo::physics::SimbodyHinge2Joint, 883
gazebo::physics::SimbodyHingeJoint, 889
gazebo::physics::SimbodyScrewJoint, 933
gazebo::physics::SimbodySliderJoint, 940
gazebo::physics::SimbodyUniversalJoint, 948
GetVelocityLimit
gazebo::physics::Joint, 510
GetVertFOV
gazebo::rendering::GpuLaser, 434
gazebo::sensors::GpuRaySensor, 446
GetVertHalfAngle
gazebo::rendering::GpuLaser, 434
gazebo::sensors::GpuRaySensor, 446
GetVertNodeWeight
gazebo::common::Skeleton, 958
GetVertex
gazebo::common::SubMesh, 1011
GetVertexCount
gazebo::common::Mesh, 611
gazebo::common::SubMesh, 1011
gazebo::physics::HeightmapShape, 469
GetVertexIndex
gazebo::common::SubMesh, 1011
GetVerticalAngleMax
gazebo::sensors::GpuRaySensor, 446
gazebo::sensors::RaySensor, 784
GetVerticalAngleMin
gazebo::sensors::GpuRaySensor, 446
gazebo::sensors::RaySensor, 784
GetVerticalMaxAngle
gazebo::physics::MultiRayShape, 670
GetVerticalMinAngle
gazebo::physics::MultiRayShape, 670
GetVerticalRangeCount
gazebo::sensors::GpuRaySensor, 447
gazebo::sensors::RaySensor, 784
GetVerticalRayCount
gazebo::sensors::GpuRaySensor, 447
gazebo::sensors::RaySensor, 784
GetVerticalSampleCount
gazebo::physics::MultiRayShape, 670
GetVerticalScanResolution
gazebo::physics::MultiRayShape, 670
GetViewControllerTypeString
gazebo::rendering::UserCamera, 1070
GetViewport
gazebo::rendering::Camera, 201
GetViewportHeight
gazebo::rendering::Camera, 201
GetViewportWidth
gazebo::rendering::Camera, 201
GetVisibilityFlags
gazebo::rendering::Visual, 1133
GetVisible
gazebo::rendering::Light, 538
gazebo::rendering::Visual, 1133
GetVisual
gazebo::rendering::Scene, 826
gazebo::rendering::UserCamera, 1070, 1071
GetVisualAt
gazebo::rendering::Scene, 826, 827
GetVisualBelow
gazebo::rendering::Scene, 827
GetVisualCount
gazebo::rendering::Scene, 827
GetVisualize
gazebo::sensors::Sensor, 844
GetVisualsBelowPoint
gazebo::rendering::Scene, 827
GetWallTime
gazebo::common::Time, 1036
gazebo::physics::State, 1000
GetWallTimeAsISOString
gazebo::common::Time, 1037
GetWidth
gazebo::common::Image, 478
gazebo::common::Video, 1116
GetWindow
gazebo::rendering::WindowManager, 1145
GetWindowId
gazebo::rendering::Camera, 201
GetWindowManager
gazebo::rendering::RenderEngine, 795
GetWorld
gazebo::physics::Base, 167
GetWorldAngularAccel
gazebo::physics::Collision, 226
gazebo::physics::Entity, 385
gazebo::physics::Link, 554
gazebo::physics::Model, 632
GetWorldAngularVel
gazebo::physics::Collision, 226
gazebo::physics::DARTLink, 319
gazebo::physics::Entity, 385
gazebo::physics::Model, 632
gazebo::physics::SimbodyLink, 905
GetWorldCFM
gazebo::physics::PhysicsEngine, 715
GetWorldCoGLinearVel
gazebo::physics::DARTLink, 319
gazebo::physics::Link, 555
gazebo::physics::SimbodyLink, 905
GetWorldCoGPose
gazebo::physics::Link, 555

GetWorldERP
gazebo::physics::PhysicsEngine, 715
GetWorldForce
gazebo::physics::DARTLink, 319
gazebo::physics::Link, 555
gazebo::physics::SimbodyLink, 905
GetWorldLinearAccel
gazebo::physics::Collision, 226
gazebo::physics::Entity, 385
gazebo::physics::Link, 555
gazebo::physics::Model, 633
GetWorldLinearVel
gazebo::physics::Collision, 226
gazebo::physics::DARTLink, 320
gazebo::physics::Entity, 385
gazebo::physics::Link, 555, 556
gazebo::physics::Model, 633
gazebo::physics::SimbodyLink, 906
GetWorldName
gazebo::sensors::Sensor, 844
GetWorldPathExtension
gazebo::common::SystemPaths, 1029
GetWorldPointOnPlane
gazebo::rendering::Camera, 202
GetWorldPose
gazebo::physics::Entity, 385
gazebo::rendering::Camera, 202
gazebo::rendering::Visual, 1133
GetWorldPosition
gazebo::rendering::Camera, 202
GetWorldRotation
gazebo::rendering::Camera, 202
GetWorldTorque
gazebo::physics::DARTLink, 320
gazebo::physics::Link, 556
gazebo::physics::SimbodyLink, 906
getWorldTransforms
gazebo::rendering::MovableText, 656
GetWorldVisual
gazebo::rendering::Scene, 827
GetWrench
gazebo::physics::LinkState, 567
GetXAxis
gazebo::math::Quaternion, 767
GetXLength
gazebo::math::Box, 174
GetYAxis
gazebo::math::Quaternion, 767
GetYLength
gazebo::math::Box, 174
GetYaw
gazebo::math::Quaternion, 767
GetZAxis
gazebo::math::Quaternion, 767
GetZLength
gazebo::math::Box, 174
GetZValue
gazebo::rendering::Camera, 202
globalEndPos
gazebo::physics::RayShape, 791
GlobalFromLocal
gazebo::common::SphericalCoordinates, 992
globalStartPos
gazebo::physics::RayShape, 791
google, 130
google::protobuf, 130
google::protobuf::compiler, 130
google::protobuf::compiler::cpp, 130
google::protobuf::compiler::cpp::GazeboGenerator, 425
∼GazeboGenerator, 426
GazeboGenerator, 426
Generate, 426
GpsSensor
gazebo::sensors::GpsSensor, 427
GpsSensor.hh, 1246
GpsSensorPtr
gazebo::sensors, 125
GpuLaser
gazebo::rendering::GpuLaser, 432
GpuLaser.hh, 1247
GpuLaserPtr
gazebo::rendering, 121
GpuRaySensor
gazebo::sensors::GpuRaySensor, 441
GpuRaySensor.hh, 1247
GpuRaySensor_V
gazebo::sensors, 125
GpuRaySensorPtr
gazebo::sensors, 125
gravity
gazebo::physics::SimbodyPhysics, 924
Green
gazebo::common::Color, 244
Grid
gazebo::rendering::Grid, 451
Grid.hh, 1248
Gripper
gazebo::physics::Gripper, 454
Gripper.hh, 1249
GripperPtr
gazebo::physics, 117
GtsSurface
MeshCSG.hh, 1283
GzTerrainMatGen
gazebo::rendering::GzTerrainMatGen, 460
gzclr_end
Common, 37
gzclr_start

Common, 37
gzdbg
Common, 37
gzerr
Common, 37
gzlog
Common, 37
gzmsg
Common, 38
gzthrow
Common, 38
gzwarn
Common, 38

H_CENTER
gazebo::rendering::MovableText, 654
H_LEFT
gazebo::rendering::MovableText, 654
HEADER_LENGTH
Connection.hh, 1211
HEIGHTMAP_SHAPE
gazebo::physics::Base, 163
HI_STOP
gazebo::physics::Joint, 500
HINGE2_JOINT
gazebo::physics::Base, 163
HINGE_JOINT
gazebo::physics::Base, 163
HalfPi
gazebo::math::Angle, 145
handle
gazebo::common::SkeletonNode, 972
gazebo::PluginT, 734
HandleData
gazebo::transport::CallbackHelper, 182
gazebo::transport::CallbackHelperT, 185
gazebo::transport::Node, 676
gazebo::transport::RawCallbackHelper, 778
gazebo::transport::SubscriptionTransport, 1019
HandleKeyPressEvent
gazebo::rendering::FPSViewController, 424
gazebo::rendering::GUIOverlay, 457
gazebo::rendering::OrbitViewController, 705
gazebo::rendering::UserCamera, 1071
gazebo::rendering::ViewController, 1119
HandleKeyReleaseEvent
gazebo::rendering::FPSViewController, 424
gazebo::rendering::GUIOverlay, 458
gazebo::rendering::OrbitViewController, 705
gazebo::rendering::UserCamera, 1071
gazebo::rendering::ViewController, 1120
HandleMessage
gazebo::transport::CallbackHelper, 183
gazebo::transport::CallbackHelperT, 185
gazebo::transport::Node, 676
gazebo::transport::RawCallbackHelper, 778
gazebo::transport::SubscriptionTransport, 1019
HandleMouseEvent
gazebo::rendering::FPSViewController, 424
gazebo::rendering::GUIOverlay, 458
gazebo::rendering::OrbitViewController, 705
gazebo::rendering::UserCamera, 1071
gazebo::rendering::ViewController, 1120
HasAttachedObject
gazebo::rendering::Visual, 1134
HasCollisionName
gazebo::util::OpenALSource, 700
HasConnections
gazebo::transport::Publisher, 759
HasJointState
gazebo::physics::ModelState, 646
HasLatchedSubscriber
gazebo::transport::Node, 676
HasLinkState
gazebo::physics::ModelState, 647
HasMesh
gazebo::common::MeshManager, 620
HasModel
Common, 43
HasModelState
gazebo::physics::WorldState, 1174
HasNode
gazebo::common::SkeletonAnimation, 963
HasSkeleton
gazebo::common::Mesh, 612
HasTransport
gazebo::transport::Publication, 753
HasType
gazebo::physics::Base, 167
HasVertex
gazebo::common::SubMesh, 1011
Heightmap
gazebo::rendering::Heightmap, 462
Heightmap.hh, 1250
HeightmapShape
gazebo::physics::HeightmapShape, 467
HeightmapShape.hh, 1251
HeightmapShapePtr
gazebo::physics, 117
heights
gazebo::physics::HeightmapShape, 470
Helpers.hh, 1252
GZ_DBL_MAX, 1254
GZ_DBL_MIN, 1254
GZ_FLT_MAX, 1254
GZ_FLT_MIN, 1254
GZ_UINT32_MAX, 1254
GZ_UINT32_MIN, 1255

hfov
gazebo::rendering::GpuLaser, 438
Hide
gazebo::rendering::GUIOverlay, 458
Hinge2Joint
gazebo::physics::Hinge2Joint, 472
Hinge2Joint.hh, 1255
HingeJoint
gazebo::physics::HingeJoint, 473
HingeJoint.hh, 1256
HorizAlign
gazebo::rendering::MovableText, 654
horzElem
gazebo::physics::MultiRayShape, 671
gazebo::sensors::GpuRaySensor, 449
horzHalfAngle
gazebo::rendering::GpuLaser, 438
horzRangeCount
gazebo::sensors::GpuRaySensor, 449
horzRayCount
gazebo::sensors::GpuRaySensor, 449

IDENTITY
gazebo::math::Matrix4, 606
IMAGE
gazebo::sensors, 126
INTERSECTION
gazebo::common::MeshCSG, 614
IOManager
gazebo::transport::IOManager, 495
IOManager.hh, 1259
id
gazebo::common::SkeletonNode, 972
gazebo::physics::TrajectoryInfo, 1061
Image
gazebo::common::Image, 476
Image.hh, 1257
imageFormat
gazebo::rendering::Camera, 211
imageHeight
gazebo::rendering::Camera, 211
imageWidth
gazebo::rendering::Camera, 211
img
gazebo::physics::HeightmapShape, 470
ImuSensor
gazebo::sensors::ImuSensor, 481
ImuSensor.hh, 1258
ImuSensor_V
gazebo::sensors, 125
ImuSensorPtr
gazebo::sensors, 125
IncCount
gazebo::transport::IOManager, 495
indexBufferCapacity
gazebo::rendering::DynamicRenderable, 378
inertiaRatio
gazebo::physics::Joint, 515
Inertial
gazebo::physics::Inertial, 485, 486
inertial
gazebo::physics::Link, 562
Inertial.hh, 1258
InertialPtr
gazebo::physics, 117
Init
Common, 43
gazebo::common::PID, 723
gazebo::Master, 584
gazebo::ModelPlugin, 641
gazebo::physics::Actor, 134
gazebo::physics::Base, 167
gazebo::physics::BoxShape, 178
gazebo::physics::Collision, 226
gazebo::physics::ContactManager, 265
gazebo::physics::CylinderShape, 278
gazebo::physics::DARTBallJoint, 283
gazebo::physics::DARTCollision, 289
gazebo::physics::DARTHeightmapShape, 294
gazebo::physics::DARTHinge2Joint, 298
gazebo::physics::DARTHingeJoint, 303
gazebo::physics::DARTJoint, 311
gazebo::physics::DARTLink, 320
gazebo::physics::DARTMeshShape, 325
gazebo::physics::DARTModel, 328
gazebo::physics::DARTPhysics, 334
gazebo::physics::DARTScrewJoint, 343
gazebo::physics::DARTSliderJoint, 348
gazebo::physics::DARTUniversalJoint, 356
gazebo::physics::Gripper, 454
gazebo::physics::HeightmapShape, 469
gazebo::physics::HingeJoint, 474
gazebo::physics::Joint, 510
gazebo::physics::Link, 556
gazebo::physics::MapShape, 582
gazebo::physics::MeshShape, 623
gazebo::physics::Model, 633
gazebo::physics::MultiRayShape, 671
gazebo::physics::PhysicsEngine, 715
gazebo::physics::PlaneShape, 731
gazebo::physics::RayShape, 789
gazebo::physics::Road, 805
gazebo::physics::Shape, 864
gazebo::physics::SimbodyBallJoint, 869
gazebo::physics::SimbodyHeightmapShape, 879
gazebo::physics::SimbodyHinge2Joint, 884
gazebo::physics::SimbodyLink, 906
gazebo::physics::SimbodyMeshShape, 911

gazebo::physics::SimbodyModel, 913
gazebo::physics::SimbodyPhysics, 920
gazebo::physics::SimbodyScrewJoint, 934
gazebo::physics::SimbodyUniversalJoint, 949
gazebo::physics::SphereShape, 988
gazebo::physics::World, 1165
gazebo::rendering::Camera, 203
gazebo::rendering::DepthCamera, 361
gazebo::rendering::DynamicRenderable, 377
gazebo::rendering::FPSViewController, 425
gazebo::rendering::GpuLaser, 434
gazebo::rendering::Grid, 452
gazebo::rendering::GUIOverlay, 458
gazebo::rendering::OrbitViewController, 705
gazebo::rendering::RenderEngine, 795
gazebo::rendering::RTShaderSystem, 813
gazebo::rendering::Scene, 828
gazebo::rendering::UserCamera, 1071
gazebo::rendering::ViewController, 1120
gazebo::rendering::Visual, 1134
gazebo::rendering::WireBox, 1147
gazebo::SensorPlugin, 855
gazebo::sensors::CameraSensor, 215
gazebo::sensors::ContactSensor, 271
gazebo::sensors::DepthCameraSensor, 363
gazebo::sensors::ForceTorqueSensor, 421
gazebo::sensors::GpsSensor, 428
gazebo::sensors::GpuRaySensor, 447
gazebo::sensors::ImuSensor, 482
gazebo::sensors::MultiCameraSensor, 663
gazebo::sensors::RaySensor, 785
gazebo::sensors::RFIDSensor, 797
gazebo::sensors::RFIDTag, 800
gazebo::sensors::Sensor, 844
gazebo::sensors::SensorManager, 852
gazebo::sensors::SonarSensor, 981
gazebo::sensors::WirelessReceiver, 1149
gazebo::sensors::WirelessTransceiver, 1152
gazebo::sensors::WirelessTransmitter, 1156
gazebo::Server, 856
gazebo::SystemPlugin, 1031
gazebo::transport::ConnectionManager, 257
gazebo::transport::Node, 677
gazebo::transport::PublicationTransport, 756
gazebo::transport::SubscribeOptions, 1016
gazebo::transport::SubscriptionTransport, 1020
gazebo::transport::TopicManager, 1059
gazebo::util::DiagnosticManager, 367
gazebo::util::LogRecord, 577
gazebo::VisualPlugin, 1143
gazebo::WorldPlugin, 1170
Messages, 60

init
gazebo, 101
Rendering, 84
Sensors, 90
Transport, 95

init_world
Classes for physics and dynamics, 70

init_worlds
Classes for physics and dynamics, 70

InitForThread
gazebo::physics::DARTPhysics, 334
gazebo::physics::PhysicsEngine, 715
gazebo::physics::SimbodyPhysics, 920

InitModel
gazebo::physics::SimbodyPhysics, 920

initialTransform
gazebo::common::SkeletonNode, 972

initialized
gazebo::rendering::Camera, 211

InsertLatchedMsg
gazebo::transport::Node, 677

InsertMesh
gazebo::rendering::Visual, 1134

InsertModelFile
gazebo::physics::World, 1165

InsertModelSDF
gazebo::physics::World, 1165

InsertModelString
gazebo::physics::World, 1165

Instance
SingletonT, 953

integ
gazebo::physics::SimbodyPhysics, 924

InternalError
gazebo::common::InternalError, 494

Interpolate
gazebo::math::RotationSpline, 808, 809
gazebo::math::Spline, 996

interpolateX
gazebo::physics::Actor, 135

invBindTransform
gazebo::common::SkeletonNode, 972

Inverse
gazebo::math::Matrix4, 602

Invert
gazebo::math::Quaternion, 768

is_stopped
Transport, 96

IsActive
gazebo::physics::Actor, 134
gazebo::sensors::CameraSensor, 216
gazebo::sensors::ContactSensor, 271
gazebo::sensors::ForceTorqueSensor, 421
gazebo::sensors::GpuRaySensor, 447
gazebo::sensors::ImuSensor, 482
gazebo::sensors::MultiCameraSensor, 663

gazebo::sensors::RaySensor, 785
gazebo::sensors::Sensor, 844
gazebo::sensors::SonarSensor, 981
IsAdvertised
gazebo::transport::TopicManager, 1059
IsAffine
gazebo::math::Matrix4, 602
IsAnimating
gazebo::rendering::Camera, 203
IsAttached
gazebo::physics::Gripper, 455
IsCanonicalLink
gazebo::physics::Entity, 386
IsFinite
gazebo::math::Pose, 739
gazebo::math::Quaternion, 768
gazebo::math::Vector2d, 1077
gazebo::math::Vector2i, 1085
gazebo::math::Vector3, 1098
gazebo::math::Vector4, 1108
IsHorizontal
gazebo::rendering::GpuLaser, 435
gazebo::sensors::GpuRaySensor, 447
isHorizontal
gazebo::rendering::GpuLaser, 438
IsInitialized
Common, 43
gazebo::rendering::GUIOverlay, 458
IsJoint
gazebo::common::SkeletonNode, 970
IsLoaded
gazebo::physics::World, 1165
IsLocal
gazebo::transport::CallbackHelper, 183
gazebo::transport::CallbackHelperT, 186
gazebo::transport::RawCallbackHelper, 778
gazebo::transport::SubscriptionTransport, 1020
IsOpen
gazebo::transport::Connection, 253
gazebo::util::LogPlay, 572
IsPaused
gazebo::physics::World, 1165
IsPlaceable
gazebo::physics::Collision, 226
IsPlane
gazebo::rendering::Visual, 1134
IsPlaying
gazebo::util::OpenALSource, 701
isPowerOfTwo
Math, 51
IsReadyToStart
gazebo::util::LogRecord, 578
IsRegistered
gazebo::physics::PhysicsFactory, 721
isReversed
gazebo::physics::SimbodyJoint, 899
IsRootNode
gazebo::common::SkeletonNode, 970
IsRunning
gazebo::transport::ConnectionManager, 257
IsSelected
gazebo::physics::Base, 168
IsStatic
gazebo::physics::Entity, 386
gazebo::rendering::Visual, 1135
IsValidFilename
gazebo::common::MeshManager, 620
IsVisible
gazebo::rendering::Camera, 203
IsZero
gazebo::physics::CollisionState, 232
gazebo::physics::JointState, 525
gazebo::physics::LinkState, 567
gazebo::physics::ModelState, 647
gazebo::physics::WorldState, 1174
isnan
Math, 51

JOINT
gazebo::common::SkeletonNode, 966
gazebo::physics::Base, 163
Joint
gazebo::physics::Joint, 500
Joint.hh, 1261
MAX_JOINT_AXIS, 1262
Joint_TEST, 516
ForceTorque1, 518
ForceTorque2, 518
GetForceTorqueWithAppliedForce, 518
Joint_TEST, 518
JointCreationDestructionTest, 518
JointTorqueTest, 519
jointType, 520
Joint_TEST, 518
physicsEngine, 520
SetUp, 519
SpawnJoint, 519
SpawnJointRotational, 520
SpawnJointRotationalWorld, 520
SpawnJointTypes, 520
Joint_TEST.hh, 1262
std_string2, 1263
Joint_TEST::SpawnJointOptions, 983
∼SpawnJointOptions, 984
axis, 984
childLinkPose, 984
jointPose, 985
modelPose, 985

noLinkPose, 985
parentLinkPose, 985
SpawnJointOptions, 984
type, 985
wait, 985
worldChild, 985
worldParent, 985
Joint_V
gazebo::physics, 117
JointController
gazebo::physics::JointController, 521
JointController.hh, 1263
JointController_V
gazebo::physics, 117
JointControllerPtr
gazebo::physics, 117
JointCreationDestructionTest
Joint_TEST, 518
jointPose
Joint_TEST::SpawnJointOptions, 985
JointPtr
gazebo::physics, 117
jointPub
gazebo::physics::Model, 638
JointState
gazebo::physics::JointState, 524
JointState.hh, 1264
JointState_M
gazebo::physics, 117
JointTorqueTest
Joint_TEST, 519
jointType
Joint_TEST, 520
JointVisual
gazebo::rendering::JointVisual, 528
JointVisual.hh, 1265
JointVisualPtr
gazebo::rendering, 121
JointWrench.hh, 1265

kd
gazebo::physics::SurfaceParams, 1023
key
gazebo::common::KeyEvent, 532
KeyEvent
gazebo::common::KeyEvent, 532
KeyEvent.hh, 1266
KeyFrame
gazebo::common::KeyFrame, 533
KeyFrame.hh, 1267
KeyFrame_V
gazebo::common::Animation, 147
keyFrames
gazebo::common::Animation, 149
gazebo::common::NodeAnimation, 683
kp
gazebo::physics::SurfaceParams, 1023

L_INT16
gazebo::common::Image, 476
L_INT8
gazebo::common::Image, 476
LEFT
gazebo::common::MouseEvent, 651
LIGHT
gazebo::physics::Base, 163
LINE_MAX_LEN
STLLoader.hh, 1379
LINES
gazebo::common::SubMesh, 1006
LINESTRIPS
gazebo::common::SubMesh, 1006
LINK
gazebo::physics::Base, 163
LINUX
SystemPaths.hh, 1387
LO_STOP
gazebo::physics::Joint, 500
Lap
gazebo::util::DiagnosticManager, 367
gazebo::util::DiagnosticTimer, 369
LaserVisual
gazebo::rendering::LaserVisual, 535
LaserVisual.hh, 1268
LaserVisualPtr
gazebo::rendering, 121
lastMeasurementTime
gazebo::sensors::Sensor, 847
lastPos
gazebo::physics::Actor, 135
lastRenderWallTime
gazebo::rendering::Camera, 211
lastScriptTime
gazebo::physics::Actor, 135
lastTraj
gazebo::physics::Actor, 135
lastUpdateTime
gazebo::sensors::Sensor, 847
latching
gazebo::transport::CallbackHelper, 183
length
gazebo::common::Animation, 149
gazebo::common::NodeAnimation, 683
gazebo::common::SkeletonAnimation, 963
Light
gazebo::rendering::Light, 537
Light.hh, 1269
LightFromSDF

Messages, 61
LightPtr
gazebo::rendering, 121
LightingModel
gazebo::rendering::RTShaderSystem, 812
limitForce
gazebo::physics::SimbodyJoint, 899
linearAccel
gazebo::physics::Link, 563
Link
gazebo::physics::Link, 547
link
gazebo::physics::Collision, 229
Link.hh, 1270
Link_V
gazebo::physics, 117
LinkPtr
gazebo::physics, 117
LinkState
gazebo::physics::LinkState, 565
LinkState.hh, 1271
LinkState_M
gazebo::physics, 117
Listen
gazebo::transport::Connection, 253
Load
gazebo::common::BVHLoader, 180
gazebo::common::ColladaLoader, 219
gazebo::common::Image, 479
gazebo::common::MeshLoader, 615
gazebo::common::MeshManager, 620
gazebo::common::STLLoader, 1003
gazebo::common::Video, 1116
gazebo::ModelPlugin, 641
gazebo::physics::Actor, 134
gazebo::physics::BallJoint, 158
gazebo::physics::Base, 168
gazebo::physics::Collision, 227
gazebo::physics::CollisionState, 232
gazebo::physics::DARTBallJoint, 283
gazebo::physics::DARTCollision, 289
gazebo::physics::DARTHinge2Joint, 298
gazebo::physics::DARTHingeJoint, 303
gazebo::physics::DARTJoint, 311
gazebo::physics::DARTLink, 321
gazebo::physics::DARTMeshShape, 325
gazebo::physics::DARTModel, 328
gazebo::physics::DARTPhysics, 334
gazebo::physics::DARTScrewJoint, 343
gazebo::physics::DARTSliderJoint, 348
gazebo::physics::DARTUniversalJoint, 356
gazebo::physics::Entity, 386
gazebo::physics::Gripper, 455
gazebo::physics::HeightmapShape, 469
gazebo::physics::Hinge2Joint, 472
gazebo::physics::HingeJoint, 474
gazebo::physics::Inertial, 488
gazebo::physics::Joint, 510
gazebo::physics::JointState, 526
gazebo::physics::Link, 556
gazebo::physics::LinkState, 568
gazebo::physics::MapShape, 582
gazebo::physics::Model, 633
gazebo::physics::ModelState, 647
gazebo::physics::PhysicsEngine, 715
gazebo::physics::Road, 805
gazebo::physics::ScrewJoint, 834
gazebo::physics::SimbodyBallJoint, 869
gazebo::physics::SimbodyCollision, 874
gazebo::physics::SimbodyHinge2Joint, 884
gazebo::physics::SimbodyHingeJoint, 889
gazebo::physics::SimbodyJoint, 897
gazebo::physics::SimbodyLink, 907
gazebo::physics::SimbodyMeshShape, 911
gazebo::physics::SimbodyModel, 913
gazebo::physics::SimbodyPhysics, 921
gazebo::physics::SimbodyScrewJoint, 934
gazebo::physics::SimbodySliderJoint, 940
gazebo::physics::SimbodyUniversalJoint, 949
gazebo::physics::SliderJoint, 975
gazebo::physics::State, 1000
gazebo::physics::SurfaceParams, 1022
gazebo::physics::UniversalJoint, 1065
gazebo::physics::World, 1166
gazebo::physics::WorldState, 1174
gazebo::rendering::ArrowVisual, 151
gazebo::rendering::AxisVisual, 156
gazebo::rendering::Camera, 203, 204
gazebo::rendering::CameraVisual, 218
gazebo::rendering::COMVisual, 246
gazebo::rendering::DepthCamera, 361
gazebo::rendering::GpuLaser, 435
gazebo::rendering::Heightmap, 463
gazebo::rendering::JointVisual, 529
gazebo::rendering::Light, 538, 539
gazebo::rendering::MovableText, 657
gazebo::rendering::Projector, 749
gazebo::rendering::RenderEngine, 795
gazebo::rendering::Road2d, 806
gazebo::rendering::Scene, 828
gazebo::rendering::SonarVisual, 983
gazebo::rendering::TransmitterVisual, 1063
gazebo::rendering::UserCamera, 1071, 1072
gazebo::rendering::Visual, 1135
gazebo::rendering::WrenchVisual, 1178
gazebo::SensorPlugin, 855
gazebo::sensors::CameraSensor, 216
gazebo::sensors::ContactSensor, 271

gazebo::sensors::DepthCameraSensor, 364
gazebo::sensors::ForceTorqueSensor, 422
gazebo::sensors::GpsSensor, 428
gazebo::sensors::GpuRaySensor, 447, 448
gazebo::sensors::ImuSensor, 482, 483
gazebo::sensors::MultiCameraSensor, 663
gazebo::sensors::Noise, 691
gazebo::sensors::RaySensor, 785
gazebo::sensors::RFIDSensor, 797
gazebo::sensors::RFIDTag, 800
gazebo::sensors::Sensor, 845
gazebo::sensors::SonarSensor, 981
gazebo::sensors::WirelessReceiver, 1150
gazebo::sensors::WirelessTransceiver, 1152
gazebo::sensors::WirelessTransmitter, 1156
gazebo::SystemPlugin, 1031
gazebo::util::OpenAL, 697
gazebo::util::OpenALSource, 701
gazebo::VisualPlugin, 1143
gazebo::WorldPlugin, 1170
Rendering, 85
load
Classes for physics and dynamics, 70
Common, 43
gazebo, 101
Rendering, 84
Sensors, 90
load_world
Classes for physics and dynamics, 70
load_worlds
Classes for physics and dynamics, 70
LoadFile
gazebo::Server, 856
LoadFromMsg
gazebo::rendering::Heightmap, 463
gazebo::rendering::Light, 539
gazebo::rendering::Visual, 1135
LoadJoints
gazebo::physics::Model, 633
LoadLayout
gazebo::rendering::GUIOverlay, 459
LoadPlugin
gazebo::physics::World, 1166
gazebo::rendering::Visual, 1135
LoadPlugins
gazebo::physics::Model, 634
loadProceduralPage
gazebo::rendering::DummyPageProvider, 370
LoadString
gazebo::Server, 856
LocalPublish
gazebo::transport::Publication, 754
Log
Common, 43
LogPlay.hh, 1273
LogRecord.hh, 1273
GZ_LOG_VERSION, 1274
Logplay, 573
loop
gazebo::common::Animation, 149
gazebo::physics::Actor, 136
Lower
gazebo::rendering::Heightmap, 464
lowerLimit
gazebo::physics::Joint, 515

m
gazebo::math::Matrix3, 599
gazebo::math::Matrix4, 606
MAP_SHAPE
gazebo::physics::Base, 163
MATRIX
gazebo::common::NodeTransform, 686
MAX_COLLIDE_RETURNS
Contact.hh, 1215
MAX_CONTACT_JOINTS
Contact.hh, 1215
MAX_CONTACTS
gazebo::physics::DARTPhysics, 332
MAX_JOINT_AXIS
Joint.hh, 1262
MESH_SHAPE
gazebo::physics::Base, 164
MIDDLE
gazebo::common::MouseEvent, 651
MIN_STEP_SIZE
gazebo::physics::DARTPhysics, 332
MODEL
gazebo::physics::Base, 163
MODEL_PLUGIN
Common, 38
MODULATE
gazebo::common::Material, 587
MOVE
gazebo::common::MouseEvent, 651
MSleep
gazebo::common::Time, 1037
MULTIRAY_SHAPE
gazebo::physics::Base, 163
mainLink
gazebo::physics::Actor, 136
mainpage.html, 1274
MakeStatic
gazebo::rendering::Visual, 1135
MapShape
gazebo::physics::MapShape, 581
MapShape.hh, 1274
Master

gazebo::Master, 584
Master.hh, 1275
masterMobod
gazebo::physics::SimbodyLink, 909
Material
gazebo::common::Material, 588
Material.hh, 1276, 1278
Math, 48
clamp, 50
equal, 50
fixnan, 50
isPowerOfTwo, 51
isnan, 51
max, 51
mean, 52
min, 52
NAN_D, 53
NAN_I, 53
parseFloat, 52
parseInt, 52
precision, 53
variance, 53
MathTypes.hh, 1278
Matrix3
gazebo::math::Matrix3, 595, 596
Matrix3.hh, 1278
Matrix4
gazebo::math::Matrix4, 601
Matrix4.hh, 1279
matter
gazebo::physics::SimbodyPhysics, 924
max
gazebo::math::Box, 176
Math, 51
maxStepSize
gazebo::physics::PhysicsEngine, 719
maxVel
gazebo::physics::SurfaceParams, 1023
mean
Math, 52
Merge
gazebo::math::Box, 174
Mesh
gazebo::common::Mesh, 608
mesh
gazebo::physics::Actor, 136
gazebo::physics::MeshShape, 624
Mesh.hh, 1280
MeshCSG
gazebo::common::MeshCSG, 614
MeshCSG.hh, 1282
GPtrArray, 1283
GtsSurface, 1283
MeshFromSDF
Messages, 61
MeshLoader
gazebo::common::MeshLoader, 615
MeshLoader.hh, 1284
MeshManager.hh, 1285
MeshShape
gazebo::physics::MeshShape, 622
MeshShape.hh, 1286
MeshShapePtr
gazebo::physics, 117
MessagePtr
gazebo::transport, 129
Messages, 54
Convert, 56–59
CreateRequest, 59
FogFromSDF, 59
GUIFromSDF, 60
GZ_REGISTER_STATIC_MSG, 56
GeometryFromSDF, 60
GetHeader, 60
Init, 60
LightFromSDF, 61
MeshFromSDF, 61
SceneFromSDF, 61
Set, 61–63
Stamp, 63
TrackVisualFromSDF, 63
VisualFromSDF, 64
MicToNano
gazebo::common::Time, 1037
MilToNano
gazebo::common::Time, 1037
min
gazebo::math::Box, 176
Math, 52
minDepth
gazebo::physics::SurfaceParams, 1023
mobod
gazebo::physics::SimbodyJoint, 900
Model
gazebo::physics::Model, 628
model
gazebo::physics::Joint, 516
Model.hh, 1287
Model_V
gazebo::physics, 117
ModelDatabase.hh, 1289
GZ_MODEL_DB_MANIFEST_FILENAME, 1289
GZ_MODEL_MANIFEST_FILENAME, 1289
modelPathsFromEnv
gazebo::common::SystemPaths, 1029
ModelPlugin
gazebo::ModelPlugin, 640
ModelPluginPtr

gazebo, 101
modelPose
Joint_TEST::SpawnJointOptions, 985
ModelPtr
gazebo::physics, 117
ModelState
gazebo::physics::ModelState, 643
ModelState.hh, 1290
ModelState_M
gazebo::physics, 117
ModelStdDesv
gazebo::sensors::WirelessTransmitter, 1156
modelTransform
gazebo::common::SkeletonNode, 972
MouseEvent
gazebo::common::MouseEvent, 651
MouseEvent.hh, 1291
MovableText
gazebo::rendering::MovableText, 655
MovableText.hh, 1293
moveScale
gazebo::common::MouseEvent, 652
MoveToPosition
gazebo::rendering::Camera, 204
gazebo::rendering::UserCamera, 1072
gazebo::rendering::Visual, 1136
MoveToPositions
gazebo::rendering::Camera, 204
gazebo::rendering::Visual, 1136
MoveToVisual
gazebo::rendering::UserCamera, 1072
Moved
gazebo::rendering::WindowManager, 1146
MsgFactory.hh, 1293
MsgFactoryFn
gazebo::msgs, 110
msgs.hh, 1294
mu1
gazebo::physics::SurfaceParams, 1024
mu2
gazebo::physics::SurfaceParams, 1024
MultiCameraSensor
gazebo::sensors::MultiCameraSensor, 661
MultiCameraSensor.hh, 1297
MultiCameraSensor_V
gazebo::sensors, 125
MultiCameraSensorPtr
gazebo::sensors, 126
MultiRayShape
gazebo::physics::MultiRayShape, 667
MultiRayShape.hh, 1298
MultiRayShapePtr
gazebo::physics, 117
mustBeBaseLink
gazebo::physics::SimbodyLink, 909
mustBreakLoopHere
gazebo::physics::SimbodyJoint, 900
mutexLastUpdateTime
gazebo::sensors::Sensor, 847

NAN_D
Math, 53
NAN_I
Math, 53
NEmpty
gazebo::sensors::WirelessTransmitter, 1156
NO_BUTTON
gazebo::common::MouseEvent, 651
NO_EVENT
gazebo::common::KeyEvent, 532
gazebo::common::MouseEvent, 651
NODE
gazebo::common::SkeletonNode, 966
NONE
gazebo::rendering::RenderEngine, 793
gazebo::sensors::Noise, 690
NObstacle
gazebo::sensors::WirelessTransmitter, 1157
NRealGen
gazebo::math, 108
NSleep
gazebo::common::Time, 1038
NULL
CommonTypes.hh, 1208
name
gazebo::common::Animation, 149
gazebo::common::Material, 593
gazebo::common::NodeAnimation, 683
gazebo::common::SkeletonAnimation, 964
gazebo::common::SkeletonNode, 972
gazebo::physics::State, 1002
gazebo::rendering::Camera, 211
near
gazebo::rendering::GpuLaser, 438
NewContact
gazebo::physics::ContactManager, 266
newData
gazebo::rendering::Camera, 211
newImageFrame
gazebo::rendering::Camera, 211
newLaserScans
gazebo::physics::MultiRayShape, 672
NewMsg
gazebo::msgs::MsgFactory, 659
NewPhysicsEngine
gazebo::physics::PhysicsFactory, 721
NewSensor
gazebo::sensors::SensorFactory, 849

noLinkPose
Joint_TEST::SpawnJointOptions, 985
Node
gazebo::transport::Node, 674
node
gazebo::physics::Entity, 389
gazebo::physics::Gripper, 455
gazebo::physics::PhysicsEngine, 719
gazebo::sensors::Sensor, 847
Node.hh, 1299
NodeAnimation
gazebo::common::NodeAnimation, 680
nodeIndex
gazebo::common::NodeAssignment, 684
NodeMap
gazebo::common, 104
NodeMapIter
gazebo::common, 104
NodePtr
gazebo::transport, 129
NodeTransform
gazebo::common::NodeTransform, 686
nodes
gazebo::common::Skeleton, 959
Noise
gazebo::sensors::Noise, 690
Noise.hh, 1300
NoisePtr
gazebo::sensors, 126
NoiseType
gazebo::sensors::Noise, 690
normal
gazebo::math::Plane, 728
NormalRealDist
gazebo::math, 108
Normalize
gazebo::math::Angle, 140
gazebo::math::Quaternion, 768
gazebo::math::Vector2d, 1077
gazebo::math::Vector2i, 1086
gazebo::math::Vector3, 1098
gazebo::math::Vector4, 1108
normals
gazebo::physics::Contact, 263
Notify
gazebo::util::LogRecord, 578
notifyRenderSingleObject
gazebo::rendering::GpuLaser, 435
nsec
gazebo::common::Time, 1052
NullStream
Common, 39
NumTerrainSubdivisions
gazebo::rendering::Heightmap, 465
NumericAnimation
gazebo::common::NumericAnimation, 692
NumericAnimationPtr
gazebo::common, 104
NumericKeyFrame
gazebo::common::NumericKeyFrame, 694

ORDER_MAX
STLLoader.hh, 1379
OTHER
gazebo::sensors, 126
offset
gazebo::physics::MultiRayShape, 672
Ogre, 130
ogre, 130
ogre_gazebo.h, 1301
ogrePathsFromEnv
gazebo::common::SystemPaths, 1029
oldAction
gazebo::physics::Actor, 136
onAnimationComplete
gazebo::rendering::Camera, 211
OnPhysicsMsg
gazebo::physics::DARTPhysics, 334
gazebo::physics::PhysicsEngine, 715
gazebo::physics::SimbodyPhysics, 921
OnPoseChange
gazebo::physics::DARTCollision, 290
gazebo::physics::DARTLink, 321
gazebo::physics::Entity, 386
gazebo::physics::Link, 557
gazebo::physics::Model, 634
gazebo::physics::SimbodyCollision, 875
gazebo::physics::SimbodyLink, 907
gazebo::rendering::Light, 539
OnRequest
gazebo::physics::DARTPhysics, 335
gazebo::physics::PhysicsEngine, 716
gazebo::physics::SimbodyPhysics, 921
One
gazebo::math::Vector3, 1104
Open
gazebo::util::LogPlay, 572
OpenAL.hh, 1302
OpenALSink
gazebo::util::OpenALSink, 698
OpenALSinkPtr
gazebo::util, 129
OpenALSource
gazebo::util::OpenALSource, 699
OpenALSourcePtr
gazebo::util, 130
operator<
gazebo::common::Time, 1045, 1046

gazebo::math::Angle, 142
operator<<
gazebo::common::Color, 243
gazebo::common::Exception, 418
gazebo::common::Material, 593
gazebo::common::Time, 1052
gazebo::common::Timer, 1054
gazebo::math::Angle, 144
gazebo::math::Box, 176
gazebo::math::Matrix3, 598
gazebo::math::Matrix4, 605
gazebo::math::Pose, 742
gazebo::math::Quaternion, 773
gazebo::math::Vector2d, 1082
gazebo::math::Vector2i, 1091
gazebo::math::Vector3, 1104
gazebo::math::Vector4, 1113
gazebo::physics::CollisionState, 233
gazebo::physics::Inertial, 492
gazebo::physics::JointState, 527
gazebo::physics::LinkState, 570
gazebo::physics::ModelState, 649
gazebo::physics::WorldState, 1176
operator<=
gazebo::common::Time, 1046, 1047
gazebo::math::Angle, 142
operator>
gazebo::common::Time, 1049
gazebo::math::Angle, 143
operator>>
gazebo::common::Color, 243
gazebo::common::Time, 1052
gazebo::math::Angle, 144
gazebo::math::Pose, 743
gazebo::math::Quaternion, 774
gazebo::math::Vector2d, 1082
gazebo::math::Vector2i, 1091
gazebo::math::Vector3, 1104
gazebo::math::Vector4, 1114
operator>=
gazebo::common::Time, 1050
gazebo::math::Angle, 143
operator*
gazebo::common::Color, 238
gazebo::common::NodeTransform, 687
gazebo::common::Time, 1039
gazebo::math::Angle, 140
gazebo::math::Matrix3, 596, 598
gazebo::math::Matrix4, 602, 603
gazebo::math::Pose, 739
gazebo::math::Quaternion, 768, 769
gazebo::math::Vector2d, 1077, 1078
gazebo::math::Vector2i, 1086
gazebo::math::Vector3, 1099, 1104
gazebo::math::Vector4, 1109
operator*=
gazebo::common::Color, 238
gazebo::common::Time, 1040
gazebo::math::Angle, 140
gazebo::math::Quaternion, 769
gazebo::math::Vector2d, 1078
gazebo::math::Vector2i, 1087
gazebo::math::Vector3, 1099
gazebo::math::Vector4, 1110
operator()
gazebo::common::NodeTransform, 687
gazebo::event::EventT, 410–412
operator+
gazebo::common::Color, 239
gazebo::common::Time, 1040, 1041
gazebo::math::Angle, 141
gazebo::math::Box, 174
gazebo::math::Matrix3, 596
gazebo::math::Pose, 740
gazebo::math::Quaternion, 769
gazebo::math::Vector2d, 1079
gazebo::math::Vector2i, 1087
gazebo::math::Vector3, 1100
gazebo::math::Vector4, 1110
gazebo::physics::CollisionState, 232
gazebo::physics::Inertial, 489
gazebo::physics::JointState, 526
gazebo::physics::JointWrench, 530
gazebo::physics::LinkState, 568
gazebo::physics::ModelState, 648
gazebo::physics::WorldState, 1175
operator+=
gazebo::common::Color, 239
gazebo::common::Time, 1041, 1042
gazebo::math::Angle, 141
gazebo::math::Box, 175
gazebo::math::Pose, 740
gazebo::math::Quaternion, 769
gazebo::math::Vector2d, 1079
gazebo::math::Vector2i, 1087
gazebo::math::Vector3, 1100
gazebo::math::Vector4, 1110
gazebo::physics::Inertial, 489
operator-
gazebo::common::Color, 239, 240
gazebo::common::Time, 1042
gazebo::math::Angle, 141
gazebo::math::Box, 175
gazebo::math::Matrix3, 597
gazebo::math::Pose, 740
gazebo::math::Quaternion, 770
gazebo::math::Vector2d, 1079
gazebo::math::Vector2i, 1088

gazebo::math::Vector3, 1100
gazebo::math::Vector4, 1111
gazebo::physics::CollisionState, 232
gazebo::physics::JointState, 526
gazebo::physics::JointWrench, 530
gazebo::physics::LinkState, 568
gazebo::physics::ModelState, 648
gazebo::physics::State, 1000
gazebo::physics::WorldState, 1175
operator-=
gazebo::common::Color, 240
gazebo::common::Time, 1043
gazebo::math::Angle, 141
gazebo::math::Pose, 741
gazebo::math::Quaternion, 770
gazebo::math::Vector2d, 1079
gazebo::math::Vector2i, 1088
gazebo::math::Vector3, 1101
gazebo::math::Vector4, 1111
operator/
gazebo::common::Color, 240
gazebo::common::Time, 1043, 1044
gazebo::math::Angle, 142
gazebo::math::Vector2d, 1079, 1080
gazebo::math::Vector2i, 1088
gazebo::math::Vector3, 1101
gazebo::math::Vector4, 1111
operator/=
gazebo::common::Color, 241
gazebo::common::Time, 1044, 1045
gazebo::math::Angle, 142
gazebo::math::Vector2d, 1080
gazebo::math::Vector2i, 1089
gazebo::math::Vector3, 1101
gazebo::math::Vector4, 1112
operator=
gazebo::common::Color, 241
gazebo::common::PID, 724
gazebo::common::Time, 1047
gazebo::math::Box, 175
gazebo::math::Matrix4, 603
gazebo::math::Plane, 727
gazebo::math::Pose, 741
gazebo::math::Quaternion, 770
gazebo::math::Vector2d, 1081
gazebo::math::Vector2i, 1089, 1090
gazebo::math::Vector3, 1102
gazebo::math::Vector4, 1112, 1113
gazebo::physics::CollisionState, 233
gazebo::physics::Contact, 262
gazebo::physics::Inertial, 489
gazebo::physics::JointState, 526
gazebo::physics::JointWrench, 530
gazebo::physics::LinkState, 569
gazebo::physics::ModelState, 648
gazebo::physics::State, 1001
gazebo::physics::WorldState, 1175
operator==
gazebo::common::Color, 241
gazebo::common::Time, 1048
gazebo::math::Angle, 143
gazebo::math::Box, 175
gazebo::math::Matrix3, 597
gazebo::math::Matrix4, 604
gazebo::math::Pose, 741
gazebo::math::Quaternion, 771
gazebo::math::Vector2d, 1081
gazebo::math::Vector2i, 1090
gazebo::math::Vector3, 1102
gazebo::math::Vector4, 1113
gazebo::physics::Base, 168
operator[]
gazebo::common::Color, 241
gazebo::math::Matrix3, 597
gazebo::math::Matrix4, 604
gazebo::math::Vector2d, 1081
gazebo::math::Vector2i, 1090
gazebo::math::Vector3, 1102
gazebo::math::Vector4, 1113
OrbitViewController
gazebo::rendering::OrbitViewController, 704
OrbitViewController.hh, 1303

PHONG
gazebo::common::Material, 588
PID
gazebo::common::PID, 723
PID.hh, 1311
PIXEL_FORMAT_COUNT
gazebo::common::Image, 476
PLANE_SHAPE
gazebo::physics::Base, 164
POINTS
gazebo::common::SubMesh, 1006
PRESS
gazebo::common::KeyEvent, 532
gazebo::common::MouseEvent, 651
Param_V
gazebo::common, 104
parent
gazebo::common::SkeletonNode, 973
gazebo::physics::Base, 171
gazebo::rendering::Visual, 1142
parentEntity
gazebo::physics::Entity, 389
gazebo::sensors::WirelessTransceiver, 1153
parentId
gazebo::sensors::Sensor, 847

parentLink
gazebo::physics::Joint, 516
parentLinkPose
Joint_TEST::SpawnJointOptions, 985
parentName
gazebo::sensors::Sensor, 847
ParseArgs
gazebo::Server, 856
parseFloat
Math, 52
parseInt
Math, 52
pathLength
gazebo::physics::Actor, 136
Pause
gazebo::util::OpenALSource, 701
pause
gazebo::event::Events, 404
pause_incoming
Transport, 96
pause_world
Classes for physics and dynamics, 71
pause_worlds
Classes for physics and dynamics, 71
PauseIncoming
gazebo::transport::TopicManager, 1059
PhysicsEngine
gazebo::physics::PhysicsEngine, 710
physicsEngine
Joint_TEST, 520
PhysicsEngine.hh, 1303
PhysicsEnginePtr
gazebo::physics, 117
PhysicsFactory.hh, 1304
PhysicsFactoryFn
Classes for physics and dynamics, 69
PhysicsIface.hh, 1306
physicsInitialized
gazebo::physics::SimbodyJoint, 900
gazebo::physics::SimbodyLink, 909
physicsSub
gazebo::physics::PhysicsEngine, 719
PhysicsTypes.hh, 1308
GZ_ALL_COLLIDE, 1310
GZ_FIXED_COLLIDE, 1310
GZ_GHOST_COLLIDE, 1310
GZ_NONE_COLLIDE, 1310
GZ_SENSOR_COLLIDE, 1310
physicsUpdateMutex
gazebo::physics::PhysicsEngine, 720
Pi
gazebo::math::Angle, 145
pitchNode
gazebo::rendering::Camera, 211
PixelFormat
gazebo::common::Image, 476
PixelFormatNames
Common, 44
PlaceOnEntity
gazebo::physics::Entity, 386
PlaceOnNearestEntityBelow
gazebo::physics::Entity, 387
placeable
gazebo::physics::Collision, 229
Plane
gazebo::math::Plane, 727
Plane.hh, 1312
PlaneShape
gazebo::physics::PlaneShape, 730
PlaneShape.hh, 1312
Play
gazebo::physics::Actor, 134
gazebo::util::OpenALSource, 701
playStartTime
gazebo::physics::Actor, 136
Plugin.hh, 1313
GZ_REGISTER_MODEL_PLUGIN, 1315
GZ_REGISTER_SENSOR_PLUGIN, 1316
GZ_REGISTER_SYSTEM_PLUGIN, 1316
GZ_REGISTER_VISUAL_PLUGIN, 1316
GZ_REGISTER_WORLD_PLUGIN, 1317
pluginPathsFromEnv
gazebo::common::SystemPaths, 1029
PluginT
gazebo::PluginT, 733
PluginType
Common, 38
plugins
gazebo::sensors::Sensor, 847
pointSize
gazebo::common::Material, 593
points
gazebo::math::RotationSpline, 810
gazebo::math::Spline, 997
pos
gazebo::common::MouseEvent, 652
gazebo::math::Pose, 743
Pose
gazebo::math::Pose, 736, 737
pose
gazebo::sensors::Sensor, 847
Pose.hh, 1317
Pose2Transform
gazebo::physics::SimbodyPhysics, 921
PoseAnimation
gazebo::common::PoseAnimation, 744
PoseAnimationPtr
gazebo::common, 104

PoseKeyFrame
gazebo::common::PoseKeyFrame, 747
poseSub
gazebo::sensors::Sensor, 847
positions
gazebo::physics::Contact, 263
PostRender
gazebo::rendering::Camera, 204
gazebo::rendering::DepthCamera, 361
gazebo::rendering::GpuLaser, 435
gazebo::rendering::UserCamera, 1072
postRender
gazebo::event::Events, 404
power
gazebo::sensors::WirelessTransceiver, 1153
PreLoad
gazebo::Server, 856
PreRender
gazebo::rendering::Scene, 828
preRender
gazebo::event::Events, 404
precision
Math, 53
PrepareHardwareBuffers
gazebo::rendering::DynamicRenderable, 378
prepareProceduralPage
gazebo::rendering::DummyPageProvider, 370
pressPos
gazebo::common::MouseEvent, 652
prevAnimTime
gazebo::rendering::Camera, 212
prevAnimationTime
gazebo::physics::Entity, 389
prevFrameTime
gazebo::physics::Actor, 136
prevPos
gazebo::common::MouseEvent, 652
PrimitiveType
gazebo::common::SubMesh, 1006
Print
gazebo::common::Exception, 418
gazebo::physics::Base, 169
print_version
gazebo, 101
PrintEntityTree
gazebo::physics::World, 1166
PrintSceneGraph
gazebo::rendering::Scene, 828
PrintSource
gazebo::common::NodeTransform, 687
PrintTransforms
gazebo::common::Skeleton, 958
PrintUsage
gazebo::Server, 856
ProcessIncoming
gazebo::transport::Node, 677
ProcessMsg
gazebo::physics::BoxShape, 178
gazebo::physics::Collision, 227
gazebo::physics::CylinderShape, 278
gazebo::physics::HeightmapShape, 470
gazebo::physics::Inertial, 489
gazebo::physics::Link, 557
gazebo::physics::MapShape, 583
gazebo::physics::MeshShape, 623
gazebo::physics::Model, 634
gazebo::physics::MultiRayShape, 671
gazebo::physics::PlaneShape, 731
gazebo::physics::RayShape, 789
gazebo::physics::Shape, 864
gazebo::physics::SphereShape, 988
gazebo::physics::SurfaceParams, 1022
ProcessNodes
gazebo::transport::TopicManager, 1059
ProcessPublishers
gazebo::transport::Node, 677
ProcessWriteQueue
gazebo::transport::Connection, 253
Projector
gazebo::rendering::Projector, 749
Projector.hh, 1318
provideFeedback
gazebo::physics::Joint, 516
pub
gazebo::sensors::WirelessTransceiver, 1153
Publication
gazebo::transport::Publication, 751
Publication.hh, 1318
PublicationPtr
gazebo::transport, 129
PublicationTransport
gazebo::transport::PublicationTransport, 755
PublicationTransport.hh, 1321
PublicationTransportPtr
gazebo::transport, 129
Publish
gazebo::transport::Node, 677
gazebo::transport::Publication, 754
gazebo::transport::Publisher, 759
gazebo::transport::TopicManager, 1059
publish
Transport, 96
PublishContacts
gazebo::physics::ContactManager, 266
PublishModelPose
gazebo::physics::World, 1166
Publisher
gazebo::transport::Publisher, 758

publisher
gazebo::physics::ContactPublisher, 267
Publisher.hh, 1323
PublisherPtr
gazebo::transport, 129
Purple
gazebo::common::Color, 244

QuadNode, 760
QuadToQuad
gazebo::physics::SimbodyPhysics, 922
Quaternion
gazebo::math::Quaternion, 764
Quaternion.hh, 1325

r
gazebo::common::Color, 244
R_FLOAT16
gazebo::common::Image, 476
R_FLOAT32
gazebo::common::Image, 476
RAY
gazebo::sensors, 126
RAY_SHAPE
gazebo::physics::Base, 164
RELEASE
gazebo::common::KeyEvent, 532
gazebo::common::MouseEvent, 651
RENDER_PATH_COUNT
gazebo::rendering::RenderEngine, 793
RENDERING_LINE_LIST
gazebo::rendering, 122
RENDERING_LINE_STRIP
gazebo::rendering, 122
RENDERING_MESH_RESOURCE
gazebo::rendering, 122
RENDERING_POINT_LIST
gazebo::rendering, 122
RENDERING_TRIANGLE_FAN
gazebo::rendering, 122
RENDERING_TRIANGLE_LIST
gazebo::rendering, 122
RENDERING_TRIANGLE_STRIP
gazebo::rendering, 122
REPLACE
gazebo::common::Material, 587
RFIDSensor
gazebo::sensors::RFIDSensor, 797
RFIDSensor.hh, 1334
RFIDSensor_V
gazebo::sensors, 126
RFIDSensorPtr
gazebo::sensors, 126
RFIDTag
gazebo::sensors::RFIDTag, 800
RFIDTag.hh, 1335
RFIDTag_V
gazebo::sensors, 126
RFIDTagPtr
gazebo::sensors, 126
RFIDTagVisual
gazebo::rendering::RFIDTagVisual, 802
RFIDTagVisual.hh, 1335
RFIDTagVisualPtr
gazebo::rendering, 121
RFIDVisual
gazebo::rendering::RFIDVisual, 803
RFIDVisual.hh, 1336
RFIDVisualPtr
gazebo::rendering, 121
RGB_FLOAT16
gazebo::common::Image, 476
RGB_FLOAT32
gazebo::common::Image, 476
RGB_INT16
gazebo::common::Image, 476
RGB_INT32
gazebo::common::Image, 476
RGB_INT8
gazebo::common::Image, 476
RGBA
gazebo::common::Color, 236
RGBA_INT8
gazebo::common::Image, 476
RIGHT
gazebo::common::MouseEvent, 651
ROT
Rendering, 83
ROT_X
Rendering, 83
ROT_Y
Rendering, 83
ROT_Z
Rendering, 83
ROTATE
gazebo::common::NodeTransform, 686
RTShaderSystem.hh, 1340
Radian
gazebo::math::Angle, 143
Raise
gazebo::rendering::Heightmap, 464
Rand.hh, 1326
rangeCountRatio
gazebo::sensors::GpuRaySensor, 449
rangeElem
gazebo::physics::MultiRayShape, 672
gazebo::sensors::GpuRaySensor, 449
RawCallbackHelper
gazebo::transport::RawCallbackHelper, 777

rawNW
gazebo::common::Skeleton, 959
RawNodeAnim
gazebo::common, 104
RawNodeWeights
gazebo::common, 104
RawSkeletonAnim
gazebo::common, 104
rawTransforms
gazebo::common::SkeletonNode, 973
rayCountRatio
gazebo::rendering::GpuLaser, 438
rayElem
gazebo::physics::MultiRayShape, 672
RaySensor
gazebo::sensors::RaySensor, 781
RaySensor.hh, 1327
RaySensor_V
gazebo::sensors, 126
RaySensorPtr
gazebo::sensors, 126
RayShape
gazebo::physics::RayShape, 788
RayShape.hh, 1328
RayShapePtr
gazebo::physics, 117
rays
gazebo::physics::MultiRayShape, 672
Read
gazebo::transport::Connection, 253
ReadCallback
gazebo::transport::Connection, 250
ReadPixelBuffer
gazebo::rendering::Camera, 205
realTime
gazebo::common::UpdateInfo, 1065
gazebo::physics::State, 1002
realTimeUpdateRate
gazebo::physics::PhysicsEngine, 720
RecalcTangents
gazebo::math::RotationSpline, 809
gazebo::math::Spline, 996
RecalculateMatrix
gazebo::common::NodeTransform, 688
RecalculateNormals
gazebo::common::Mesh, 612
gazebo::common::SubMesh, 1012
Red
gazebo::common::Color, 244
referencePose
gazebo::sensors::WirelessTransceiver, 1153
RegisterAll
gazebo::physics::PhysicsFactory, 721
gazebo::sensors::SensorFactory, 849
RegisterMsg
gazebo::msgs::MsgFactory, 659
RegisterPhysicsEngine
gazebo::physics::PhysicsFactory, 721
RegisterSensor
gazebo::sensors::SensorFactory, 849
RegisterTopicNamespace
gazebo::transport::ConnectionManager, 257
gazebo::transport::TopicManager, 1060
relativeEndPos
gazebo::physics::RayShape, 791
relativeStartPos
gazebo::physics::RayShape, 791
Remove
gazebo::util::LogRecord, 578
remove_scene
Rendering, 85
remove_sensor
Sensors, 90
remove_sensors
Sensors, 91
remove_worlds
Classes for physics and dynamics, 71
RemoveCallback
gazebo::transport::Node, 678
RemoveCamera
gazebo::rendering::Scene, 828
RemoveChild
gazebo::physics::Base, 169
gazebo::physics::Link, 557
gazebo::physics::Model, 634
RemoveChildJoint
gazebo::physics::Link, 557
RemoveChildren
gazebo::physics::Base, 169
RemoveCollision
gazebo::physics::Link, 557
RemoveConnection
gazebo::transport::ConnectionManager, 257
RemoveNode
gazebo::transport::TopicManager, 1060
RemoveParentJoint
gazebo::physics::Link, 557
RemovePlugin
gazebo::physics::World, 1166
gazebo::rendering::Visual, 1136
RemoveProjectors
gazebo::rendering::Scene, 828
RemoveScene
gazebo::rendering::RenderEngine, 795
gazebo::rendering::RTShaderSystem, 813
removeScene
gazebo::rendering::Events, 407
RemoveSensor

gazebo::sensors::SensorManager, 852
RemoveSensors
gazebo::sensors::SensorManager, 852
RemoveShadows
gazebo::rendering::RTShaderSystem, 814
RemoveSubscription
gazebo::transport::Publication, 754
RemoveTransport
gazebo::transport::Publication, 754
RemoveVisual
gazebo::rendering::Scene, 828
Render
gazebo::rendering::Camera, 205
render
gazebo::event::Events, 404
RenderEngine.hh, 1329
RenderEvents.hh, 1330
RenderImpl
gazebo::rendering::Camera, 205
RenderOpType
gazebo::rendering, 122
RenderPathType
gazebo::rendering::RenderEngine, 793
renderTarget
gazebo::rendering::Camera, 212
renderTexture
gazebo::rendering::Camera, 212
RenderTypes.hh, 1332
GZ_VISIBILITY_ALL, 1333
GZ_VISIBILITY_GUI, 1333
GZ_VISIBILITY_SELECTABLE, 1334
GZ_VISIBILITY_SELECTION, 1334
Rendering, 80
∼SelectionObj, 83
Attach, 83
create_scene, 84
Detach, 84
fini, 84
get_scene, 84
GetMode, 84
GetState, 84
init, 84
Load, 85
load, 84
ROT, 83
ROT_X, 83
ROT_Y, 83
ROT_Z, 83
remove_scene, 85
SCALE, 83
SCALE_X, 83
SCALE_Y, 83
SCALE_Z, 83
SELECTION_NONE, 83
SelectionMode, 83
SelectionObj, 83
SetGlobal, 85
SetMode, 85
SetState, 85
TRANS, 83
TRANS_X, 83
TRANS_Y, 83
TRANS_Z, 83
UpdateSize, 86
RenderingIface.hh, 1330
request
Transport, 96
requestNoReply
Transport, 96, 97
requestPub
gazebo::physics::Entity, 390
requestSub
gazebo::physics::PhysicsEngine, 720
requests
gazebo::rendering::Camera, 212
Rescale
gazebo::common::Image, 479
Reset
gazebo::common::Color, 242
gazebo::common::PID, 724
gazebo::common::SkeletonNode, 970
gazebo::math::Pose, 741
gazebo::ModelPlugin, 641
gazebo::physics::Base, 169
gazebo::physics::Contact, 262
gazebo::physics::DARTJoint, 311
gazebo::physics::DARTPhysics, 335
gazebo::physics::Entity, 387
gazebo::physics::Inertial, 490
gazebo::physics::Joint, 511
gazebo::physics::JointController, 521
gazebo::physics::Link, 557
gazebo::physics::Model, 634
gazebo::physics::PhysicsEngine, 716
gazebo::physics::SimbodyJoint, 897
gazebo::physics::SimbodyPhysics, 922
gazebo::physics::World, 1167
gazebo::SensorPlugin, 855
gazebo::SystemPlugin, 1031
gazebo::VisualPlugin, 1143
gazebo::WorldPlugin, 1170
ResetCount
gazebo::physics::ContactManager, 266
ResetEntities
gazebo::physics::World, 1167
ResetLastUpdateTime
gazebo::sensors::Sensor, 845
ResetLastUpdateTimes

gazebo::sensors::SensorManager, 852
ResetPhysicsStates
gazebo::physics::Link, 558
ResetTime
gazebo::physics::World, 1167
Resize
gazebo::rendering::GUIOverlay, 459
gazebo::rendering::UserCamera, 1072
gazebo::rendering::WindowManager, 1146
responsePub
gazebo::physics::PhysicsEngine, 720
RestoreSimbodyState
gazebo::physics::SimbodyHingeJoint, 890
gazebo::physics::SimbodyJoint, 897
gazebo::physics::SimbodyLink, 907
RestoreState
gazebo::physics::DARTModel, 328
Rewind
gazebo::util::OpenALSource, 701
Road, 804
gazebo::physics::Road, 805
Road.hh, 1337
Road2d
gazebo::rendering::Road2d, 806
Road2d.hh, 1338
RoadPtr
gazebo::physics, 117
root
gazebo::common::Skeleton, 959
gazebo::rendering::RenderEngine, 795
rot
gazebo::math::Pose, 743
Rotate
gazebo::physics::Inertial, 490
rotate
gazebo::common::PoseKeyFrame, 748
RotatePitch
gazebo::rendering::Camera, 205
RotatePositionAboutOrigin
gazebo::math::Pose, 741
RotateVector
gazebo::math::Quaternion, 771
RotateVectorReverse
gazebo::math::Quaternion, 771
RotateYaw
gazebo::rendering::Camera, 205
RotationSpline
gazebo::math::RotationSpline, 807
RotationSpline.hh, 1338
Round
gazebo::math::Pose, 742
gazebo::math::Quaternion, 771
gazebo::math::Vector3, 1103
Run
gazebo::Master, 584
gazebo::physics::World, 1167
gazebo::Server, 856
gazebo::transport::ConnectionManager, 257
run
gazebo, 101
Transport, 97
run_once
Sensors, 91
run_threads
Sensors, 91
run_world
Classes for physics and dynamics, 71
run_worlds
Classes for physics and dynamics, 71
RunOnce
gazebo::Master, 584
RunThread
gazebo::Master, 584
RunThreads
gazebo::sensors::SensorManager, 853

SCALE
gazebo::common::NodeTransform, 686
Rendering, 83
SCALE_X
Rendering, 83
SCALE_Y
Rendering, 83
SCALE_Z
Rendering, 83
SCREW_JOINT
gazebo::physics::Base, 163
SCROLL
gazebo::common::MouseEvent, 651
SELECTION_NONE
Rendering, 83
SENSOR_COLLISION
gazebo::physics::Base, 164
SENSOR_PLUGIN
Common, 38
SHADE_COUNT
gazebo::common::Material, 588
SHAPE
gazebo::physics::Base, 163
SLIDER_JOINT
gazebo::physics::Base, 163
SM2Profile
gazebo::rendering::GzTerrainMatGen::SM2Profile, 977
SPHERE_SHAPE
gazebo::physics::Base, 164
SSLM_NormalMapLightingObjectSpace
gazebo::rendering::RTShaderSystem, 812

SSLM_NormalMapLightingTangentSpace
gazebo::rendering::RTShaderSystem, 812
SSLM_PerPixelLighting
gazebo::rendering::RTShaderSystem, 812
SSLM_PerVertexLighting
gazebo::rendering::RTShaderSystem, 812
STLLoader
gazebo::common::STLLoader, 1003
STLLoader.hh, 1378
COR3_MAX, 1379
FACE_MAX, 1379
LINE_MAX_LEN, 1379
ORDER_MAX, 1379
STOP_CFM
gazebo::physics::Joint, 500
STOP_ERP
gazebo::physics::Joint, 500
SUSPENSION_CFM
gazebo::physics::Joint, 500
SUSPENSION_ERP
gazebo::physics::Joint, 500
SYSTEM_PLUGIN
Common, 38
Save
gazebo::physics::World, 1167
saveCount
gazebo::rendering::Camera, 212
SaveFrame
gazebo::rendering::Camera, 205, 206
gazebo::sensors::CameraSensor, 216
gazebo::sensors::DepthCameraSensor, 364
gazebo::sensors::MultiCameraSensor, 664
saveFrameBuffer
gazebo::rendering::Camera, 212
SavePNG
gazebo::common::Image, 479
SaveSimbodyState
gazebo::physics::SimbodyHingeJoint, 890
gazebo::physics::SimbodyJoint, 897
gazebo::physics::SimbodyLink, 907
Scale
gazebo::common::Mesh, 612
gazebo::common::NodeAnimation, 682
gazebo::common::Skeleton, 958
gazebo::common::SkeletonAnimation, 963
gazebo::common::SubMesh, 1012
gazebo::math::Quaternion, 771
scale
gazebo::physics::Entity, 390
gazebo::physics::Shape, 865
ScaleXAxis
gazebo::rendering::AxisVisual, 156
ScaleYAxis
gazebo::rendering::AxisVisual, 156
ScaleZAxis
gazebo::rendering::AxisVisual, 156
scanElem
gazebo::physics::MultiRayShape, 672
gazebo::sensors::GpuRaySensor, 449
Scene
gazebo::rendering::Scene, 819
scene
gazebo::rendering::Camera, 212
gazebo::rendering::Visual, 1142
gazebo::sensors::Sensor, 848
Scene.hh, 1340
SceneFromSDF
Messages, 61
sceneNode
gazebo::rendering::Camera, 212
gazebo::rendering::Visual, 1142
ScenePtr
gazebo::rendering, 121
screenshotPath
gazebo::rendering::Camera, 212
ScrewJoint
gazebo::physics::ScrewJoint, 833
ScrewJoint.hh, 1342
scriptLength
gazebo::physics::Actor, 136
scroll
gazebo::common::MouseEvent, 652
sdf
gazebo::physics::Base, 171
gazebo::physics::PhysicsEngine, 720
gazebo::rendering::Camera, 212
gazebo::sensors::Sensor, 848
sec
gazebo::common::Time, 1052
SecToNano
gazebo::common::Time, 1051
SelectVisual
gazebo::rendering::Scene, 829
SelectionMode
Rendering, 83
SelectionObj
Rendering, 83
SelectionObj.hh, 1343
SelectionObjPtr
gazebo::rendering, 122
SendMessage
gazebo::transport::Publisher, 759
Sensor
gazebo::sensors::Sensor, 840
Sensor.hh, 1343
Sensor_V
gazebo::sensors, 126
SensorCategory

gazebo::sensors, 126
SensorFactor, 848
SensorFactory.hh, 1344
SensorFactoryFn
gazebo::sensors, 126
SensorManager.hh, 1345
SensorPlugin
gazebo::SensorPlugin, 854
SensorPluginPtr
gazebo, 101
SensorPtr
gazebo::sensors, 126
SensorTypes.hh, 1348
Sensors, 87
create_sensor, 89
disable, 89
enable, 90
fini, 90
GZ_REGISTER_STATIC_SENSOR, 89
get_sensor, 90
init, 90
load, 90
remove_sensor, 90
remove_sensors, 91
run_once, 91
run_threads, 91
stop, 91
SensorsIface.hh, 1346
SensorsInitialized
gazebo::sensors::SensorManager, 853
Server
gazebo::Server, 856
Server.hh, 1350
Set
gazebo::common::Color, 242
gazebo::common::NodeTransform, 688
gazebo::common::Time, 1051
gazebo::math::Matrix4, 604
gazebo::math::Plane, 728
gazebo::math::Pose, 742
gazebo::math::Quaternion, 772
gazebo::math::Vector2d, 1082
gazebo::math::Vector2i, 1090
gazebo::math::Vector3, 1103
gazebo::math::Vector4, 1113
Messages, 61–63
SetActive
gazebo::sensors::DepthCameraSensor, 364
gazebo::sensors::Sensor, 845
SetAltitude
gazebo::physics::DARTPlaneShape, 337
gazebo::physics::PlaneShape, 731
gazebo::physics::SimbodyPlaneShape, 926
SetAmbient
gazebo::common::Material, 590
gazebo::rendering::Visual, 1136
SetAmbientColor
gazebo::rendering::Scene, 829
SetAnchor
gazebo::physics::DARTJoint, 311
gazebo::physics::Joint, 511
gazebo::physics::ScrewJoint, 834
gazebo::physics::SimbodyJoint, 897
gazebo::physics::SliderJoint, 975
SetAngle
gazebo::physics::Joint, 511
SetAngleMax
gazebo::sensors::GpuRaySensor, 448
SetAngleMin
gazebo::sensors::GpuRaySensor, 448
SetAngularAccel
gazebo::physics::Link, 558
gazebo::physics::Model, 634
SetAngularDamping
gazebo::physics::DARTLink, 321
gazebo::physics::Link, 558
gazebo::physics::SimbodyLink, 907
SetAngularVel
gazebo::physics::DARTLink, 321
gazebo::physics::Link, 558
gazebo::physics::Model, 634
gazebo::physics::SimbodyLink, 907
SetAnimation
gazebo::physics::Entity, 387
SetAspectRatio
gazebo::rendering::Camera, 206
SetAttenuation
gazebo::rendering::Light, 539
SetAttribute
gazebo::physics::DARTJoint, 311
gazebo::physics::Joint, 511
gazebo::physics::SimbodyJoint, 897, 898
SetAutoCalculate
gazebo::math::RotationSpline, 809
gazebo::math::Spline, 996
SetAutoDisable
gazebo::physics::DARTLink, 321
gazebo::physics::Link, 558
gazebo::physics::Model, 635
gazebo::physics::SimbodyLink, 907
SetAutoDisableFlag
gazebo::physics::PhysicsEngine, 716
SetAxis
gazebo::physics::BallJoint, 159
gazebo::physics::DARTHinge2Joint, 298
gazebo::physics::DARTHingeJoint, 303
gazebo::physics::DARTScrewJoint, 343
gazebo::physics::DARTSliderJoint, 348

gazebo::physics::DARTUniversalJoint, 356
gazebo::physics::Joint, 512
gazebo::physics::SimbodyHinge2Joint, 884
gazebo::physics::SimbodyHingeJoint, 890
gazebo::physics::SimbodyJoint, 898
gazebo::physics::SimbodyScrewJoint, 934
gazebo::physics::SimbodySliderJoint, 940
gazebo::physics::SimbodyUniversalJoint, 949
SetAxisMaterial
gazebo::rendering::AxisVisual, 157
SetBackgroundColor
gazebo::rendering::Scene, 829
SetBasePath
gazebo::util::LogRecord, 578
SetBaseline
gazebo::rendering::MovableText, 657
SetBindShapeTransform
gazebo::common::Skeleton, 958
SetBlendFactors
gazebo::common::Material, 591
SetBlendMode
gazebo::common::Material, 591
SetCallbackId
gazebo::transport::Subscriber, 1017
SetCamera
gazebo::rendering::WindowManager, 1146
SetCameraCount
gazebo::rendering::GpuLaser, 435
SetCanonicalLink
gazebo::physics::Entity, 387
SetCaptureData
gazebo::rendering::Camera, 206
SetCaptureDataOnce
gazebo::rendering::Camera, 206
SetCastShadows
gazebo::rendering::Light, 539
gazebo::rendering::Visual, 1136
SetCategoryBits
gazebo::physics::Collision, 227
gazebo::physics::DARTCollision, 290
gazebo::physics::SimbodyCollision, 875
SetCellCount
gazebo::rendering::Grid, 452
SetCellLength
gazebo::rendering::Grid, 452
SetCharHeight
gazebo::rendering::MovableText, 657
SetClipDist
gazebo::rendering::Camera, 206
SetCmd
gazebo::common::PID, 724
SetCmdMax
gazebo::common::PID, 724
SetCmdMin
gazebo::common::PID, 724
SetCoG
gazebo::physics::Inertial, 490
SetCol
gazebo::math::Matrix3, 597
SetCollideBits
gazebo::physics::Collision, 227
gazebo::physics::DARTCollision, 290
gazebo::physics::SimbodyCollision, 875
SetCollideMode
gazebo::physics::Link, 558
gazebo::physics::Model, 635
SetCollision
gazebo::physics::Collision, 227
SetCollisionShape
gazebo::physics::SimbodyCollision, 875
SetColor
gazebo::rendering::DynamicLines, 374
gazebo::rendering::Grid, 453
gazebo::rendering::MovableText, 657
SetComponent
gazebo::common::NodeTransform, 688
SetContactMaxCorrectingVel
gazebo::physics::PhysicsEngine, 716
SetContactSurfaceLayer
gazebo::physics::PhysicsEngine, 716
SetContactsEnabled
gazebo::physics::Collision, 228
SetCosHorzFOV
gazebo::rendering::GpuLaser, 435
SetCosVertFOV
gazebo::rendering::GpuLaser, 435
SetDARTCollisionShape
gazebo::physics::DARTCollision, 290
SetDARTParentJoint
gazebo::physics::DARTLink, 321
SetDGain
gazebo::common::PID, 725
SetDamping
gazebo::physics::DARTJoint, 312
gazebo::physics::Joint, 512
gazebo::physics::SimbodyBallJoint, 869
gazebo::physics::SimbodyHinge2Joint, 884
gazebo::physics::SimbodyHingeJoint, 890
gazebo::physics::SimbodyJoint, 898
gazebo::physics::SimbodyScrewJoint, 934
gazebo::physics::SimbodySliderJoint, 941
gazebo::physics::SimbodyUniversalJoint, 949
SetDepthTarget
gazebo::rendering::DepthCamera, 361
SetDepthWrite
gazebo::common::Material, 591
SetDiffuse
gazebo::common::Material, 591

gazebo::rendering::Visual, 1136
SetDiffuseColor
gazebo::rendering::Light, 539
SetDirection
gazebo::rendering::Light, 539
SetDirtyPose
gazebo::physics::SimbodyLink, 908
SetDistance
gazebo::rendering::OrbitViewController, 706
SetEffortLimit
gazebo::physics::Joint, 512
SetElevationReference
gazebo::common::SphericalCoordinates, 992
SetEmissive
gazebo::common::Material, 591
gazebo::rendering::LaserVisual, 535
gazebo::rendering::Visual, 1137
SetEnabled
gazebo::physics::DARTLink, 322
gazebo::physics::Link, 559
gazebo::physics::Model, 635
gazebo::physics::SimbodyLink, 908
gazebo::rendering::ContactVisual, 273
gazebo::rendering::Projector, 750
gazebo::rendering::ViewController, 1120
gazebo::rendering::WrenchVisual, 1178
SetFarClip
gazebo::rendering::GpuLaser, 436
SetFiducial
gazebo::physics::RayShape, 790
SetFile
gazebo::common::AudioDecoder, 154
SetFocalPoint
gazebo::rendering::OrbitViewController, 706
gazebo::rendering::UserCamera, 1073
SetFog
gazebo::rendering::Scene, 829
SetFontName
gazebo::rendering::MovableText, 657
SetForce
gazebo::physics::DARTJoint, 312
gazebo::physics::DARTLink, 322
gazebo::physics::Joint, 512
gazebo::physics::Link, 559
gazebo::physics::SimbodyJoint, 898
gazebo::physics::SimbodyLink, 908
SetForceImpl
gazebo::physics::DARTBallJoint, 283
gazebo::physics::DARTHinge2Joint, 298
gazebo::physics::DARTHingeJoint, 303
gazebo::physics::DARTJoint, 312
gazebo::physics::DARTScrewJoint, 343
gazebo::physics::DARTSliderJoint, 349
gazebo::physics::DARTUniversalJoint, 356
gazebo::physics::SimbodyBallJoint, 869
gazebo::physics::SimbodyHinge2Joint, 884
gazebo::physics::SimbodyHingeJoint, 890
gazebo::physics::SimbodyJoint, 899
gazebo::physics::SimbodyScrewJoint, 934
gazebo::physics::SimbodySliderJoint, 941
gazebo::physics::SimbodyUniversalJoint, 949
SetFromABGR
gazebo::common::Color, 242
SetFromARGB
gazebo::common::Color, 242
SetFromAxes
gazebo::math::Matrix3, 598
SetFromAxis
gazebo::math::Matrix3, 598
gazebo::math::Quaternion, 772
SetFromBGRA
gazebo::common::Color, 242
SetFromData
gazebo::common::Image, 479
SetFromDegree
gazebo::math::Angle, 144
SetFromEuler
gazebo::math::Quaternion, 772
SetFromHSV
gazebo::common::Color, 242
SetFromRGBA
gazebo::common::Color, 243
SetFromRadian
gazebo::math::Angle, 144
SetFromYUV
gazebo::common::Color, 243
SetGain
gazebo::util::OpenALSource, 701
SetGlobal
Rendering, 85
SetGravity
gazebo::physics::DARTPhysics, 335
gazebo::physics::PhysicsEngine, 717
gazebo::physics::SimbodyPhysics, 922
SetGravityMode
gazebo::physics::DARTLink, 322
gazebo::physics::Link, 559
gazebo::physics::Model, 635
gazebo::physics::SimbodyLink, 908
SetGrid
gazebo::rendering::Scene, 829
SetHFOV
gazebo::rendering::Camera, 207
SetHandle
gazebo::common::SkeletonNode, 970
SetHeadingOffset
gazebo::common::SphericalCoordinates, 992
SetHeight

gazebo::rendering::Grid, 453
SetHighStop
gazebo::physics::BallJoint, 159
gazebo::physics::DARTJoint, 313
gazebo::physics::Joint, 513
gazebo::physics::SimbodyBallJoint, 870
gazebo::physics::SimbodyHinge2Joint, 884
gazebo::physics::SimbodyHingeJoint, 891
gazebo::physics::SimbodyScrewJoint, 935
gazebo::physics::SimbodySliderJoint, 941
gazebo::physics::SimbodyUniversalJoint, 950
SetHighlighted
gazebo::rendering::Visual, 1137
SetHorzFOV
gazebo::rendering::GpuLaser, 436
SetHorzHalfAngle
gazebo::rendering::GpuLaser, 436
SetIGain
gazebo::common::PID, 725
SetIMax
gazebo::common::PID, 725
SetIMin
gazebo::common::PID, 725
SetIXX
gazebo::physics::Inertial, 491
SetIXY
gazebo::physics::Inertial, 491
SetIXZ
gazebo::physics::Inertial, 491
SetIYY
gazebo::physics::Inertial, 491
SetIYZ
gazebo::physics::Inertial, 492
SetIZZ
gazebo::physics::Inertial, 492
SetId
gazebo::common::SkeletonNode, 970
gazebo::rendering::Visual, 1137
SetImageHeight
gazebo::rendering::Camera, 207
SetImageSize
gazebo::rendering::Camera, 207
SetImageWidth
gazebo::rendering::Camera, 207
SetIndexCount
gazebo::common::SubMesh, 1012
SetInertiaMatrix
gazebo::physics::Inertial, 491
SetInertial
gazebo::physics::Link, 559
SetInitialRelativePose
gazebo::physics::Entity, 387
SetInitialTransform
gazebo::common::SkeletonNode, 971
SetInverseBindTransform
gazebo::common::SkeletonNode, 971
SetIsHorizontal
gazebo::rendering::GpuLaser, 436
SetJointAnimation
gazebo::physics::Model, 635
SetJointPosition
gazebo::physics::JointController, 522
gazebo::physics::Model, 635
SetJointPositions
gazebo::physics::JointController, 522
gazebo::physics::Model, 636
SetKinematic
gazebo::physics::DARTLink, 322
gazebo::physics::Link, 559
SetLaserRetro
gazebo::physics::Collision, 228
gazebo::physics::Link, 560
gazebo::physics::Model, 636
SetLatitudeReference
gazebo::common::SphericalCoordinates, 992
SetLength
gazebo::common::Animation, 148
gazebo::physics::CylinderShape, 278
gazebo::physics::RayShape, 790
SetLightType
gazebo::rendering::Light, 540
SetLighting
gazebo::common::Material, 592
SetLineWidth
gazebo::rendering::Grid, 453
SetLinearAccel
gazebo::physics::Link, 560
gazebo::physics::Model, 636
SetLinearDamping
gazebo::physics::DARTLink, 322
gazebo::physics::Link, 560
gazebo::physics::SimbodyLink, 908
SetLinearVel
gazebo::physics::DARTLink, 323
gazebo::physics::Link, 560
gazebo::physics::Model, 636
gazebo::physics::SimbodyLink, 908
SetLinkStatic
gazebo::physics::DARTLink, 323
gazebo::physics::Link, 560
gazebo::physics::SimbodyLink, 909
SetLinkWorldPose
gazebo::physics::Model, 636, 637
SetLocallyAdvertised
gazebo::transport::Publication, 754
SetLongitudeReference
gazebo::common::SphericalCoordinates, 992
SetLoop

gazebo::util::OpenALSource, 702
SetLowStop
gazebo::physics::BallJoint, 159
gazebo::physics::DARTJoint, 313
gazebo::physics::Joint, 513
gazebo::physics::SimbodyBallJoint, 870
gazebo::physics::SimbodyHinge2Joint, 885
gazebo::physics::SimbodyHingeJoint, 891
gazebo::physics::SimbodyScrewJoint, 935
gazebo::physics::SimbodySliderJoint, 941
gazebo::physics::SimbodyUniversalJoint, 950
SetMOI
gazebo::physics::Inertial, 492
SetMass
gazebo::physics::Inertial, 492
SetMaterial
gazebo::rendering::Visual, 1137
SetMaterialIndex
gazebo::common::SubMesh, 1012
SetMaxContacts
gazebo::physics::Collision, 228
gazebo::physics::PhysicsEngine, 717
SetMaxForce
gazebo::physics::DARTBallJoint, 283
gazebo::physics::DARTHinge2Joint, 299
gazebo::physics::DARTHingeJoint, 304
gazebo::physics::DARTScrewJoint, 344
gazebo::physics::DARTSliderJoint, 349
gazebo::physics::DARTUniversalJoint, 357
gazebo::physics::Joint, 513
gazebo::physics::SimbodyBallJoint, 870
gazebo::physics::SimbodyHinge2Joint, 885
gazebo::physics::SimbodyHingeJoint, 891
gazebo::physics::SimbodyScrewJoint, 935
gazebo::physics::SimbodySliderJoint, 942
gazebo::physics::SimbodyUniversalJoint, 950
SetMaxStepSize
gazebo::physics::PhysicsEngine, 717
SetMesh
gazebo::physics::MeshShape, 623
setMinimalComms
Transport, 97
SetMode
Rendering, 85
SetModel
gazebo::physics::Joint, 514
SetModelTransform
gazebo::common::SkeletonNode, 971
SetName
gazebo::common::Mesh, 612
gazebo::common::NodeAnimation, 683
gazebo::common::SkeletonAnimation, 963
gazebo::common::SkeletonNode, 971
gazebo::common::SubMesh, 1012
gazebo::physics::Base, 170
gazebo::physics::Entity, 387
gazebo::physics::State, 1001
gazebo::rendering::Camera, 207
gazebo::rendering::Light, 540
gazebo::rendering::Visual, 1137
SetNearClip
gazebo::rendering::GpuLaser, 436
SetNode
gazebo::transport::Publisher, 760
SetNormal
gazebo::common::SubMesh, 1012
gazebo::physics::PlaneShape, 731
SetNormalCount
gazebo::common::SubMesh, 1013
SetNormalMap
gazebo::rendering::Visual, 1137
SetNumVertAttached
gazebo::common::Skeleton, 959
SetOperationType
gazebo::rendering::DynamicRenderable, 378
SetPGain
gazebo::common::PID, 725
SetParam
gazebo::physics::PhysicsEngine, 717
SetParams
gazebo::Server, 856
SetParent
gazebo::common::SkeletonNode, 971
gazebo::physics::Base, 170
gazebo::sensors::Sensor, 845, 846
SetPath
gazebo::common::Mesh, 612
SetPaused
gazebo::physics::World, 1167
gazebo::util::LogRecord, 578
SetPerPixelLighting
gazebo::rendering::RTShaderSystem, 814
SetPitch
gazebo::util::OpenALSource, 702
SetPoint
gazebo::rendering::DynamicLines, 374
SetPointSize
gazebo::common::Material, 592
SetPoints
DART Physics, 75
gazebo::physics::RayShape, 790
gazebo::physics::SimbodyRayShape, 928
SetPose
gazebo::rendering::Visual, 1138
gazebo::util::OpenALSink, 698
gazebo::util::OpenALSource, 702
SetPosition
gazebo::rendering::Light, 540

gazebo::rendering::Visual, 1138
SetPrimitiveType
gazebo::common::SubMesh, 1013
SetProvideFeedback
gazebo::physics::Joint, 514
SetPublication
gazebo::transport::Publisher, 760
SetPublishData
gazebo::physics::Link, 560
SetQuiet
Common, 43
SetRadius
gazebo::physics::CylinderShape, 278
gazebo::physics::DARTSphereShape, 351
gazebo::physics::SimbodySphereShape, 944
gazebo::physics::SphereShape, 988
SetRange
gazebo::rendering::Light, 540
SetRangeCount
gazebo::rendering::GpuLaser, 436
SetRayCountRatio
gazebo::rendering::GpuLaser, 437
SetRealTime
gazebo::physics::LinkState, 569
gazebo::physics::ModelState, 648
gazebo::physics::State, 1001
gazebo::physics::WorldState, 1175
SetRealTimeUpdateRate
gazebo::physics::PhysicsEngine, 717
SetReferencePose
gazebo::sensors::ImuSensor, 483
SetRelativePose
gazebo::physics::Entity, 388
SetRenderRate
gazebo::rendering::Camera, 207
SetRenderTarget
gazebo::rendering::Camera, 208
gazebo::rendering::UserCamera, 1073
SetRetro
gazebo::physics::RayShape, 790
SetRibbonTrail
gazebo::rendering::Visual, 1138
SetRootNode
gazebo::common::Skeleton, 959
SetRotation
gazebo::common::PoseKeyFrame, 747
gazebo::rendering::Visual, 1138
SetSID
gazebo::common::NodeTransform, 688
SetSORPGSIters
gazebo::physics::PhysicsEngine, 718
SetSORPGSPreconIters
gazebo::physics::PhysicsEngine, 718
SetSORPGSW
gazebo::physics::PhysicsEngine, 718
SetSaveFramePathname
gazebo::rendering::Camera, 208
SetSaveable
gazebo::physics::Base, 170
SetScale
gazebo::common::Mesh, 612
gazebo::common::SubMesh, 1013
gazebo::math::Matrix4, 605
gazebo::physics::BoxShape, 179
gazebo::physics::Collision, 228
gazebo::physics::CylinderShape, 278
gazebo::physics::HeightmapShape, 470
gazebo::physics::Link, 561
gazebo::physics::MapShape, 583
gazebo::physics::MeshShape, 623
gazebo::physics::Model, 637
gazebo::physics::MultiRayShape, 671
gazebo::physics::PlaneShape, 732
gazebo::physics::RayShape, 790
gazebo::physics::Shape, 864
gazebo::physics::SphereShape, 988
gazebo::rendering::Visual, 1138
SetScene
gazebo::rendering::Camera, 208
gazebo::rendering::Visual, 1139
SetSceneNode
gazebo::rendering::Camera, 208
SetSeed
gazebo::math::Rand, 776
gazebo::physics::DARTPhysics, 335
gazebo::physics::PhysicsEngine, 718
gazebo::physics::SimbodyPhysics, 922
SetSelected
gazebo::physics::Base, 170
gazebo::physics::Link, 561
gazebo::rendering::Light, 540
setSelectedEntity
gazebo::event::Events, 404
SetSelfCollide
gazebo::physics::DARTLink, 323
gazebo::physics::Link, 561
gazebo::physics::SimbodyLink, 909
SetShadeMode
gazebo::common::Material, 592
SetShaderType
gazebo::rendering::Visual, 1139
SetShadowsEnabled
gazebo::rendering::Scene, 830
SetShape
gazebo::physics::Collision, 228
SetShininess
gazebo::common::Material, 592
SetShowOnTop

gazebo::rendering::MovableText, 657
SetSimTime
gazebo::physics::LinkState, 569
gazebo::physics::ModelState, 648
gazebo::physics::State, 1001
gazebo::physics::World, 1167
gazebo::physics::WorldState, 1176
SetSize
gazebo::physics::BoxShape, 179
gazebo::physics::CylinderShape, 278
gazebo::physics::DARTBoxShape, 286
gazebo::physics::DARTCylinderShape, 292
gazebo::physics::PlaneShape, 732
gazebo::physics::SimbodyBoxShape, 872
gazebo::physics::SimbodyCylinderShape, 877
SetSkeleton
gazebo::common::Mesh, 612
SetSkeletonPose
gazebo::rendering::Visual, 1139
SetSkyXMode
gazebo::rendering::Scene, 830
SetSourceValues
gazebo::common::NodeTransform, 688
SetSpaceWidth
gazebo::rendering::MovableText, 658
SetSpecular
gazebo::common::Material, 592
gazebo::rendering::Visual, 1139
SetSpecularColor
gazebo::rendering::Light, 540
SetSpotFalloff
gazebo::rendering::Light, 541
SetSpotInnerAngle
gazebo::rendering::Light, 541
SetSpotOuterAngle
gazebo::rendering::Light, 541
SetState
gazebo::physics::Collision, 229
gazebo::physics::Joint, 514
gazebo::physics::Link, 561
gazebo::physics::Model, 637
gazebo::physics::World, 1168
Rendering, 85
SetStatic
gazebo::physics::Entity, 388
SetSubMeshCenter
gazebo::common::SubMesh, 1013
SetSurfaceType
gazebo::common::SphericalCoordinates, 993
SetTargetRealTimeFactor
gazebo::physics::PhysicsEngine, 718
SetTension
gazebo::math::Spline, 996
SetTexCoord
gazebo::common::SubMesh, 1013
SetTexCoordCount
gazebo::common::SubMesh, 1013
SetText
gazebo::rendering::MovableText, 658
SetTextAlignment
gazebo::rendering::MovableText, 658
SetTexture
gazebo::rendering::Projector, 750
SetTextureImage
gazebo::common::Material, 592
SetThreadPitch
gazebo::physics::DARTScrewJoint, 344
gazebo::physics::ScrewJoint, 835
gazebo::physics::SimbodyScrewJoint, 935
SetTime
gazebo::common::Animation, 148
SetToIdentity
gazebo::math::Quaternion, 773
SetToMax
gazebo::math::Vector3, 1103
SetToMin
gazebo::math::Vector3, 1103
SetToWallTime
gazebo::common::Time, 1051
SetTorque
gazebo::physics::DARTLink, 323
gazebo::physics::Link, 561
gazebo::physics::SimbodyLink, 909
SetTransform
gazebo::common::SkeletonNode, 971
SetTranslate
gazebo::math::Matrix4, 605
SetTranslation
gazebo::common::PoseKeyFrame, 747
SetTransparency
gazebo::common::Material, 593
gazebo::rendering::Visual, 1139
SetTransparent
gazebo::rendering::Scene, 830
SetType
gazebo::common::NodeTransform, 689
gazebo::common::SkeletonNode, 972
SetUp
Joint_TEST, 519
SetUpdateRate
gazebo::sensors::Sensor, 846
SetUserData
gazebo::rendering::Grid, 453
SetValue
gazebo::common::NumericKeyFrame, 695
SetVelocity
gazebo::physics::DARTBallJoint, 284
gazebo::physics::DARTHinge2Joint, 299

gazebo::physics::DARTHingeJoint, 304
gazebo::physics::DARTScrewJoint, 344
gazebo::physics::DARTSliderJoint, 349
gazebo::physics::DARTUniversalJoint, 357
gazebo::physics::Joint, 514
gazebo::physics::SimbodyBallJoint, 870
gazebo::physics::SimbodyHinge2Joint, 885
gazebo::physics::SimbodyHingeJoint, 891
gazebo::physics::SimbodyScrewJoint, 936
gazebo::physics::SimbodySliderJoint, 942
gazebo::physics::SimbodyUniversalJoint, 950
gazebo::util::OpenALSink, 698
gazebo::util::OpenALSource, 702
SetVertFOV
gazebo::rendering::GpuLaser, 437
SetVertHalfAngle
gazebo::rendering::GpuLaser, 437
SetVertex
gazebo::common::SubMesh, 1014
SetVertexCount
gazebo::common::SubMesh, 1014
SetVerticalAngleMax
gazebo::sensors::GpuRaySensor, 448
SetVerticalAngleMin
gazebo::sensors::GpuRaySensor, 448
SetViewController
gazebo::rendering::UserCamera, 1073
SetViewportDimensions
gazebo::rendering::UserCamera, 1073
SetVisibilityFlags
gazebo::rendering::Visual, 1139
SetVisible
gazebo::rendering::Scene, 830
gazebo::rendering::Visual, 1140
gazebo::rendering::WireBox, 1147
SetWallTime
gazebo::physics::LinkState, 569
gazebo::physics::ModelState, 649
gazebo::physics::State, 1002
gazebo::physics::WorldState, 1176
SetWindowId
gazebo::rendering::Camera, 208
SetWireframe
gazebo::rendering::Heightmap, 464
gazebo::rendering::Scene, 830
gazebo::rendering::Visual, 1140
SetWorld
gazebo::physics::Base, 170
gazebo::physics::WorldState, 1176
SetWorldCFM
gazebo::physics::PhysicsEngine, 719
SetWorldERP
gazebo::physics::PhysicsEngine, 719
SetWorldPose
gazebo::physics::Entity, 388
gazebo::rendering::Camera, 208
gazebo::rendering::UserCamera, 1074
gazebo::rendering::Visual, 1140
SetWorldPosition
gazebo::rendering::Camera, 209
gazebo::rendering::Visual, 1140
SetWorldRotation
gazebo::rendering::Camera, 209
gazebo::rendering::Visual, 1140
SetWorldTwist
gazebo::physics::Entity, 388
ShadeMode
gazebo::common::Material, 587
shadeMode
gazebo::common::Material, 594
ShadeModeStr
gazebo::common::Material, 594
Shape
gazebo::physics::Shape, 863
shape
gazebo::physics::Collision, 229
Shape.hh, 1351
ShapePtr
gazebo::physics, 117
shift
gazebo::common::MouseEvent, 652
shininess
gazebo::common::Material, 594
Show
gazebo::rendering::GUIOverlay, 459
ShowBoundingBox
gazebo::rendering::Visual, 1141
ShowCOM
gazebo::rendering::Visual, 1141
ShowCOMs
gazebo::rendering::Scene, 831
ShowClouds
gazebo::rendering::Scene, 830
ShowCollision
gazebo::rendering::Visual, 1141
ShowCollisions
gazebo::rendering::Scene, 831
ShowContacts
gazebo::rendering::Scene, 831
ShowJoints
gazebo::rendering::Scene, 831
gazebo::rendering::Visual, 1141
ShowRotation
gazebo::rendering::ArrowVisual, 151
gazebo::rendering::AxisVisual, 157
ShowSkeleton
gazebo::rendering::Visual, 1141
ShowVisual

gazebo::rendering::Light, 541
ShowWireframe
gazebo::rendering::Camera, 209
Shutdown
gazebo::transport::Connection, 254
sid
gazebo::common::NodeTransform, 689
sigInt
gazebo::event::Events, 404
Signal
gazebo::event::EventT, 413–415
SimTK, 130
simTime
gazebo::common::UpdateInfo, 1065
gazebo::physics::State, 1002
Simbody Physics, 78
simbody_inc.h, 1352
SimbodyBallJoint
gazebo::physics::SimbodyBallJoint, 867
SimbodyBallJoint.hh, 1353
SimbodyBoxShape
gazebo::physics::SimbodyBoxShape, 872
SimbodyBoxShape.hh, 1354
SimbodyCollision
gazebo::physics::SimbodyCollision, 874
SimbodyCollision.hh, 1354
SimbodyCollisionPtr
gazebo::physics, 117
SimbodyCylinderShape
gazebo::physics::SimbodyCylinderShape, 877
SimbodyCylinderShape.hh, 1355
SimbodyHeightmapShape
gazebo::physics::SimbodyHeightmapShape, 879
SimbodyHeightmapShape.hh, 1355
SimbodyHinge2Joint
gazebo::physics::SimbodyHinge2Joint, 881
SimbodyHinge2Joint.hh, 1356
SimbodyHingeJoint
gazebo::physics::SimbodyHingeJoint, 887
SimbodyHingeJoint.hh, 1357
SimbodyJoint
gazebo::physics::SimbodyJoint, 894
SimbodyJoint.hh, 1357
SimbodyLink
gazebo::physics::SimbodyLink, 903
SimbodyLink.hh, 1358
SimbodyLinkPtr
gazebo::physics, 117
SimbodyMeshShape
gazebo::physics::SimbodyMeshShape, 911
SimbodyMeshShape.hh, 1359
SimbodyModel
gazebo::physics::SimbodyModel, 913
SimbodyModel.hh, 1359
SimbodyModelPtr
gazebo::physics, 117
SimbodyMultiRayShape
gazebo::physics::SimbodyMultiRayShape, 915
SimbodyMultiRayShape.hh, 1360
SimbodyPhysics
gazebo::physics::SimbodyPhysics, 918
simbodyPhysics
gazebo::physics::SimbodyJoint, 900
SimbodyPhysics.hh, 1360
simbodyPhysicsInitialized
gazebo::physics::SimbodyPhysics, 924
SimbodyPhysicsPtr
gazebo::physics, 118
simbodyPhysicsStepped
gazebo::physics::SimbodyPhysics, 924
SimbodyPlaneShape
gazebo::physics::SimbodyPlaneShape, 926
SimbodyPlaneShape.hh, 1361
SimbodyRayShape
gazebo::physics::SimbodyRayShape, 928
SimbodyRayShape.hh, 1362
SimbodyRayShapePtr
gazebo::physics, 118
SimbodyScrewJoint
gazebo::physics::SimbodyScrewJoint, 931
SimbodyScrewJoint.hh, 1363
SimbodySliderJoint
gazebo::physics::SimbodySliderJoint, 938
SimbodySliderJoint.hh, 1363
SimbodySphereShape
gazebo::physics::SimbodySphereShape, 943
SimbodySphereShape.hh, 1364
SimbodyTypes.hh, 1364
SimbodyUniversalJoint
gazebo::physics::SimbodyUniversalJoint, 946
SimbodyUniversalJoint.hh, 1366
SingletonT
∼SingletonT, 953
Instance, 953
SingletonT, 953
SingletonT, 953
SingletonT< T >, 951
SingletonT.hh, 1366
size
gazebo::math::Plane, 728
skelAnimation
gazebo::physics::Actor, 136
skelNodesMap
gazebo::physics::Actor, 136
Skeleton
gazebo::common::Skeleton, 955
skeleton
gazebo::physics::Actor, 136

Skeleton.hh, 1367
SkeletonAnimation
gazebo::common::SkeletonAnimation, 961
SkeletonAnimation.hh, 1369
SkeletonNode
gazebo::common::SkeletonNode, 966, 967
SkeletonNodeType
gazebo::common::SkeletonNode, 966
skinFile
gazebo::physics::Actor, 137
skinScale
gazebo::physics::Actor, 137
SkyX, 130
SkyXMode
gazebo::rendering::Scene, 818
skyx
gazebo::rendering::Scene, 832
slaveMobods
gazebo::physics::SimbodyLink, 909
slaveWelds
gazebo::physics::SimbodyLink, 909
Sleep
gazebo::common::Time, 1051
Slerp
gazebo::math::Quaternion, 773
SliderJoint
gazebo::physics::SliderJoint, 974
SliderJoint.hh, 1370
slip1
gazebo::physics::SurfaceParams, 1024
slip2
gazebo::physics::SurfaceParams, 1024
Smooth
gazebo::rendering::Heightmap, 464
SnapVisualToNearestBelow
gazebo::rendering::Scene, 831
SonarSensor
gazebo::sensors::SonarSensor, 979
SonarSensor.hh, 1371
SonarSensorPtr
gazebo::sensors, 126
SonarVisual
gazebo::rendering::SonarVisual, 983
SonarVisual.hh, 1372
SonarVisualPtr
gazebo::rendering, 122
source
gazebo::common::NodeTransform, 689
SpawnJoint
Joint_TEST, 519
SpawnJointOptions
Joint_TEST::SpawnJointOptions, 984
SpawnJointRotational
Joint_TEST, 520
SpawnJointRotationalWorld
Joint_TEST, 520
SpawnJointTypes
Joint_TEST, 520
specular
gazebo::common::Material, 594
SpeedOfLight
gazebo::common, 105
SphereShape
gazebo::physics::SphereShape, 987
SphereShape.hh, 1373
SphereShapePtr
gazebo::physics, 118
SphericalCoordinates
gazebo::common::SphericalCoordinates, 990
SphericalCoordinates.hh, 1374
SphericalCoordinatesPtr
gazebo::common, 104
SphericalFromLocal
gazebo::common::SphericalCoordinates, 993
Spline
gazebo::math::Spline, 994
Spline.hh, 1375
SplitHeights
gazebo::rendering::Heightmap, 465
Squad
gazebo::math::Quaternion, 773
Stamp
Messages, 63
Start
Common, 44
gazebo::common::Timer, 1054
gazebo::util::DiagnosticTimer, 369
gazebo::util::LogRecord, 579
startDelay
gazebo::physics::Actor, 137
StartRead
gazebo::transport::Connection, 254
startTime
gazebo::physics::TrajectoryInfo, 1061
StartTimer
gazebo::util::DiagnosticManager, 367
State
gazebo::physics::State, 999
State.hh, 1377
std_string2
Joint_TEST.hh, 1263
Step
gazebo::util::LogPlay, 573
step
gazebo::event::Events, 404
StepWorld
gazebo::physics::World, 1168
Stop

gazebo::common::Timer, 1054
gazebo::Master, 585
gazebo::physics::Actor, 134
gazebo::physics::World, 1168
gazebo::sensors::SensorManager, 853
gazebo::Server, 856
gazebo::transport::ConnectionManager, 258
gazebo::transport::IOManager, 495
gazebo::util::DiagnosticTimer, 369
gazebo::util::LogRecord, 579
gazebo::util::OpenALSource, 703
stop
gazebo, 101
gazebo::event::Events, 405
Sensors, 91
Transport, 97
stop_world
Classes for physics and dynamics, 71
stop_worlds
Classes for physics and dynamics, 72
StopAnimation
gazebo::physics::Entity, 389
gazebo::physics::Model, 637
StopRead
gazebo::transport::Connection, 254
StopTimer
gazebo::util::DiagnosticManager, 367
StrStr_M
gazebo::common, 105
StripSceneName
gazebo::rendering::Scene, 831
StripWorldName
gazebo::physics::World, 1168
SubMesh
gazebo::common::SubMesh, 1006
SubNodeMap
gazebo::transport::TopicManager, 1056
subSampling
gazebo::physics::HeightmapShape, 470
submesh
gazebo::physics::MeshShape, 624
Subscribe
gazebo::transport::ConnectionManager, 258
gazebo::transport::Node, 678, 679
gazebo::transport::TopicManager, 1060
SubscribeOptions
gazebo::transport::SubscribeOptions, 1015
SubscribeOptions.hh, 1379
Subscriber
gazebo::transport::Subscriber, 1017
Subscriber.hh, 1381
SubscriberPtr
gazebo::transport, 129
SubscriptionTransport
gazebo::transport::SubscriptionTransport, 1019
SubscriptionTransport.hh, 1383
SubscriptionTransportPtr
gazebo::transport, 129
SurfaceParams
gazebo::physics::SurfaceParams, 1021
SurfaceParams.hh, 1385
SurfaceParamsPtr
gazebo::physics, 118
SurfaceType
gazebo::common::SphericalCoordinates, 990
system
gazebo::physics::SimbodyPhysics, 924
SystemPaths.hh, 1386
GetCurrentDir, 1387
LINUX, 1387
SystemPlugin
gazebo::SystemPlugin, 1030
SystemPluginPtr
gazebo, 101
systemPluginsArgc
gazebo::Server, 856
systemPluginsArgv
gazebo::Server, 857

TPtr
gazebo::PluginT, 733
TRANS
Rendering, 83
TRANS_X
Rendering, 83
TRANS_Y
Rendering, 83
TRANS_Z
Rendering, 83
TRANSLATE
gazebo::common::NodeTransform, 686
TRIANGLES
gazebo::common::SubMesh, 1006
TRIFANS
gazebo::common::SubMesh, 1006
TRISTRIPS
gazebo::common::SubMesh, 1006
tangents
gazebo::math::RotationSpline, 810
gazebo::math::Spline, 997
targetRealTimeFactor
gazebo::physics::PhysicsEngine, 720
tension
gazebo::math::Spline, 997
texImage
gazebo::common::Material, 594
textureHeight
gazebo::rendering::Camera, 212

textureWidth
gazebo::rendering::Camera, 212
threadPitch
gazebo::physics::ScrewJoint, 835
Time
gazebo::common::Time, 1035, 1036
time
gazebo::common::KeyFrame, 533
gazebo::physics::Contact, 263
Time.hh, 1387
timePos
gazebo::common::Animation, 149
Timer
gazebo::common::Timer, 1053
Timer.hh, 1388
Toggle
gazebo::rendering::Projector, 750
ToggleShowVisual
gazebo::rendering::Light, 541
ToggleShowWireframe
gazebo::rendering::Camera, 209
ToggleVisible
gazebo::rendering::Visual, 1141
TopicManager.hh, 1389
TrackVisual
gazebo::rendering::Camera, 209
TrackVisualFromSDF
Messages, 63
TrackVisualImpl
gazebo::rendering::Camera, 209, 210
gazebo::rendering::UserCamera, 1074
tracker
gazebo::physics::SimbodyPhysics, 924
trajInfo
gazebo::physics::Actor, 137
trajectories
gazebo::physics::Actor, 137
transform
gazebo::common::NodeTransform, 689
gazebo::common::SkeletonNode, 973
Transform2Pose
gazebo::physics::SimbodyPhysics, 923
TransformAffine
gazebo::math::Matrix4, 605
TransformType
gazebo::common::NodeTransform, 686
Translate
gazebo::common::Mesh, 613
gazebo::common::SubMesh, 1014
gazebo::rendering::Camera, 210
translate
gazebo::common::PoseKeyFrame, 748
translated
gazebo::physics::TrajectoryInfo, 1061
TransmitterVisual
gazebo::rendering::TransmitterVisual, 1063
TransmitterVisual.hh, 1391
transparency
gazebo::common::Material, 594
Transport, 92
CallbackHelperPtr, 94
clear_buffers, 94
fini, 94
get_master_uri, 94
get_topic_namespaces, 94
getAdvertisedTopics, 95
getMinimalComms, 95
getTopicMsgType, 95
init, 95
is_stopped, 96
pause_incoming, 96
publish, 96
request, 96
requestNoReply, 96, 97
run, 97
setMinimalComms, 97
stop, 97
TransportIface.hh, 1392
TransportTypes.hh, 1394
TriggerUpdate
gazebo::transport::ConnectionManager, 258
TwoPi
gazebo::math::Angle, 145
type
gazebo::common::KeyEvent, 532
gazebo::common::MouseEvent, 652
gazebo::common::NodeTransform, 689
gazebo::common::SkeletonNode, 973
gazebo::physics::TrajectoryInfo, 1061
gazebo::PluginT, 734
Joint_TEST::SpawnJointOptions, 985
typeString
gazebo::rendering::ViewController, 1121

UIntGen
gazebo::math, 108
UNION
gazebo::common::MeshCSG, 614
UNIVERSAL_JOINT
gazebo::physics::Base, 163
UNKNOWN_PIXEL_FORMAT
gazebo::common::Image, 476
URealGen
gazebo::math, 108
Unadvertise
gazebo::transport::ConnectionManager, 258
gazebo::transport::TopicManager, 1060
UniformIntDist

gazebo::math, 108
UniformRealDist
gazebo::math, 108
UnitX
gazebo::math::Vector3, 1104
UnitY
gazebo::math::Vector3, 1104
UnitZ
gazebo::math::Vector3, 1105
UniversalJoint
gazebo::physics::UniversalJoint, 1064
UniversalJoint.hh, 1395
unloadProceduralPage
gazebo::rendering::DummyPageProvider, 371
unprepareProceduralPage
gazebo::rendering::DummyPageProvider, 371
Unsubscribe
gazebo::transport::ConnectionManager, 258
gazebo::transport::Subscriber, 1017
gazebo::transport::TopicManager, 1060
Update
DART Physics, 75
gazebo::common::PID, 725
gazebo::physics::Actor, 135
gazebo::physics::Base, 171
gazebo::physics::DARTMeshShape, 325
gazebo::physics::DARTModel, 328
gazebo::physics::Joint, 514
gazebo::physics::JointController, 522
gazebo::physics::Link, 562
gazebo::physics::MapShape, 583
gazebo::physics::MeshShape, 624
gazebo::physics::Model, 637
gazebo::physics::MultiRayShape, 671
gazebo::physics::RayShape, 790
gazebo::physics::SimbodyRayShape, 929
gazebo::rendering::Camera, 210
gazebo::rendering::DynamicLines, 374
gazebo::rendering::FPSViewController, 425
gazebo::rendering::GUIOverlay, 459
gazebo::rendering::MovableText, 658
gazebo::rendering::OrbitViewController, 706
gazebo::rendering::TransmitterVisual, 1063
gazebo::rendering::UserCamera, 1074
gazebo::rendering::ViewController, 1121
gazebo::rendering::Visual, 1141
gazebo::sensors::Sensor, 846
gazebo::sensors::SensorManager, 853
update
gazebo::sensors::ForceTorqueSensor, 422
gazebo::sensors::SonarSensor, 982
UpdateChildrenTransforms
gazebo::common::SkeletonNode, 972
UpdateCollision
gazebo::physics::DARTPhysics, 335
gazebo::physics::PhysicsEngine, 719
gazebo::physics::SimbodyPhysics, 923
updateDirtyPoseFromDARTTransformation
gazebo::physics::DARTLink, 323
UpdateFromMsg
gazebo::rendering::Light, 541
gazebo::rendering::Visual, 1141
UpdateImpl
gazebo::sensors::CameraSensor, 216
gazebo::sensors::ContactSensor, 271
gazebo::sensors::DepthCameraSensor, 364
gazebo::sensors::ForceTorqueSensor, 422
gazebo::sensors::GpsSensor, 428
gazebo::sensors::GpuRaySensor, 449
gazebo::sensors::ImuSensor, 483
gazebo::sensors::MultiCameraSensor, 664
gazebo::sensors::RaySensor, 785
gazebo::sensors::RFIDSensor, 798
gazebo::sensors::RFIDTag, 800
gazebo::sensors::Sensor, 846
gazebo::sensors::SonarSensor, 981
gazebo::sensors::WirelessTransmitter, 1156
UpdateInfo.hh, 1396
UpdateMass
gazebo::physics::Link, 562
UpdateParameters
gazebo::physics::Actor, 135
gazebo::physics::Base, 171
gazebo::physics::Collision, 229
gazebo::physics::Entity, 389
gazebo::physics::Inertial, 492
gazebo::physics::Joint, 514
gazebo::physics::Link, 562
gazebo::physics::Model, 637
UpdateParams
gazebo::rendering::GzTerrainMatGen::SM2Profile, 977
updateParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 861
UpdateParamsForCompositeMap
gazebo::rendering::GzTerrainMatGen::SM2Profile, 977
updatePeriod
gazebo::sensors::Sensor, 848
UpdatePhysics
gazebo::physics::DARTPhysics, 335
gazebo::physics::PhysicsEngine, 719
gazebo::physics::SimbodyPhysics, 923
UpdatePoint
gazebo::math::RotationSpline, 809
gazebo::math::Spline, 997
UpdatePublications

gazebo::transport::TopicManager, 1061
UpdateRays
Bullet Physics, 77
gazebo::physics::MultiRayShape, 671
gazebo::physics::SimbodyMultiRayShape, 915
UpdateShaders
gazebo::rendering::RTShaderSystem, 814
UpdateSize
Rendering, 86
UpdateStateSDF
gazebo::physics::World, 1168
UpdateSurface
gazebo::physics::Link, 562
updateVpParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 861
upperLimit
gazebo::physics::Joint, 516
useCFMDamping
gazebo::physics::Joint, 516
UserCamera
gazebo::rendering::UserCamera, 1068
UserCamera.hh, 1397
UserCameraPtr
gazebo::rendering, 122
UtilTypes.hh, 1398
Utility, 98
DIAG_TIMER_LAP, 98
DIAG_TIMER_START, 98
DIAG_TIMER_STOP, 98

V_ABOVE
gazebo::rendering::MovableText, 655
V_BELOW
gazebo::rendering::MovableText, 655
VEL
gazebo::physics::Joint, 500
VERTEX
gazebo::rendering::RenderEngine, 793
VISUAL
gazebo::physics::Base, 163
VISUAL_PLUGIN
Common, 38
Valid
gazebo::common::Image, 479
ValidateIP
gazebo::transport::Connection, 254
value
gazebo::common::NumericKeyFrame, 695
variance
Math, 53
Vec3ToVector3
gazebo::physics::SimbodyPhysics, 923
Vector2d
gazebo::math::Vector2d, 1076
Vector2d.hh, 1399
Vector2i
gazebo::math::Vector2i, 1084, 1085
Vector2i.hh, 1400
Vector3
gazebo::math::Vector3, 1094, 1095
Vector3.hh, 1401
Vector3ToVec3
gazebo::physics::SimbodyPhysics, 923
Vector4
gazebo::math::Vector4, 1107
Vector4.hh, 1401
velocityLimit
gazebo::physics::Joint, 516
VertAlign
gazebo::rendering::MovableText, 654
vertElem
gazebo::physics::MultiRayShape, 672
gazebo::sensors::GpuRaySensor, 449
vertHalfAngle
gazebo::rendering::GpuLaser, 438
vertRangeCount
gazebo::sensors::GpuRaySensor, 449
vertRayCount
gazebo::sensors::GpuRaySensor, 450
vertSize
gazebo::physics::HeightmapShape, 470
vertexBufferCapacity
gazebo::rendering::DynamicRenderable, 378
vertexIndex
gazebo::common::NodeAssignment, 684
vfov
gazebo::rendering::GpuLaser, 438
Video
gazebo::common::Video, 1115
Video.hh, 1403
VideoVisual
gazebo::rendering::VideoVisual, 1117
VideoVisual.hh, 1404
ViewController
gazebo::rendering::ViewController, 1119
ViewController.hh, 1405
viewport
gazebo::rendering::Camera, 213
visPub
gazebo::physics::Entity, 390
visitRenderables
gazebo::rendering::MovableText, 658
Visual
gazebo::rendering::Visual, 1127
Visual.hh, 1406
VisualFromSDF
Messages, 64

visualId
gazebo::physics::Actor, 137
visualMsg
gazebo::physics::Entity, 390
visualName
gazebo::physics::Actor, 137
VisualPlugin
gazebo::VisualPlugin, 1143
VisualPluginPtr
gazebo, 101
VisualPtr
gazebo::rendering, 122
visuals
gazebo::physics::Link, 563
Visuals_M
gazebo::physics::Link, 547

w
gazebo::math::Quaternion, 774
gazebo::math::Vector4, 1114
WORLD_PLUGIN
Common, 38
wait
Joint_TEST::SpawnJointOptions, 985
WaitForConnection
gazebo::transport::Publisher, 760
wallTime
gazebo::physics::State, 1002
weight
gazebo::common::NodeAssignment, 684
White
gazebo::common::Color, 244
windowId
gazebo::rendering::Camera, 213
WindowManager
gazebo::rendering::WindowManager, 1144
WindowManager.hh, 1407
WindowManagerPtr
gazebo::rendering, 122
WireBox
gazebo::rendering::WireBox, 1147
WireBox.hh, 1408
WirelessReceiver
gazebo::sensors::WirelessReceiver, 1149
WirelessReceiver.hh, 1409
WirelessReceiver_V
gazebo::sensors, 126
WirelessReceiverPtr
gazebo::sensors, 126
WirelessTransceiver
gazebo::sensors::WirelessTransceiver, 1151
WirelessTransceiver.hh, 1409
WirelessTransceiver_V
gazebo::sensors, 126
WirelessTransceiverPtr
gazebo::sensors, 126
WirelessTransmitter
gazebo::sensors::WirelessTransmitter, 1155
WirelessTransmitter.hh, 1410
WirelessTransmitter_V
gazebo::sensors, 126
WirelessTransmitterPtr
gazebo::sensors, 126
World
gazebo::physics::World, 1160
world
gazebo::physics::Base, 171
gazebo::physics::Contact, 263
gazebo::physics::PhysicsEngine, 720
gazebo::physics::SimbodyJoint, 900
gazebo::sensors::Sensor, 848
World.hh, 1411
worldChild
Joint_TEST::SpawnJointOptions, 985
worldCreated
gazebo::event::Events, 405
worldName
gazebo::common::UpdateInfo, 1066
worldParent
Joint_TEST::SpawnJointOptions, 985
WorldPlugin
gazebo::WorldPlugin, 1170
WorldPluginPtr
gazebo, 101
WorldPtr
gazebo::physics, 118
WorldState
gazebo::physics::WorldState, 1172
WorldState.hh, 1412
worldUpdateBegin
gazebo::event::Events, 405
worldUpdateEnd
gazebo::event::Events, 405
worlds_running
Classes for physics and dynamics, 72
wrench
gazebo::physics::Contact, 263
gazebo::physics::Joint, 516
WrenchVisual
gazebo::rendering::WrenchVisual, 1177
WrenchVisual.hh, 1414
WrenchVisualPtr
gazebo::rendering, 122
Write
gazebo::util::LogRecord, 579

x
gazebo::math::Quaternion, 774

gazebo::math::Vector2d, 1082
gazebo::math::Vector2i, 1091
gazebo::math::Vector3, 1105
gazebo::math::Vector4, 1114
X_POSITION
BVHLoader.hh, 1196
X_ROTATION
BVHLoader.hh, 1196
xCB
gazebo::physics::SimbodyJoint, 900
xPA
gazebo::physics::SimbodyJoint, 900

y
gazebo::math::Quaternion, 774
gazebo::math::Vector2d, 1082
gazebo::math::Vector2i, 1091
gazebo::math::Vector3, 1105
gazebo::math::Vector4, 1114
Y_POSITION
BVHLoader.hh, 1196
Y_ROTATION
BVHLoader.hh, 1196
Yellow
gazebo::common::Color, 244

z
gazebo::math::Quaternion, 774
gazebo::math::Vector3, 1105
gazebo::math::Vector4, 1114
Z_POSITION
BVHLoader.hh, 1196
Z_ROTATION
BVHLoader.hh, 1196
ZERO
gazebo::math::Matrix4, 606
Zero
gazebo::common::Time, 1052
gazebo::math::Angle, 145
gazebo::math::Pose, 743
gazebo::math::Vector3, 1105